BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

BBN Report No. 3438

SPEECH UNDERSTANDING SYSTEMS

Final Report

November 1964 - October 1976

Volume IV. Syntax and Semantics

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: BBN ~~Report No.~~ 3438 Vol 1-5	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SPEECH UNDERSTANDING SYSTEMS. ~~Final Technical Progress Report 30 October 1974 to 29 October 1976~~	5. ~~TYPE OF REPORT~~ & PERIOD COVERED: Final Tech. Prog. Report 30 Oct 1974 to 29 Oct 1976
	6. PERFORMING ORG. REPORT NUMBER: BBN Report No. 3438
7. AUTHOR(s): W. Woods, M. Bates, G. Brown, B. Bruce, C. Cook, J. Klovstad, J. Makhoul, B. Nash-Webber, R. Schwartz, J. Wolf, V. Zue.	8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0533,
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Bolt Beranek and Newman Inc. 50 Moulton Street Cambridge, Mass. 02138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 5D30
11. CONTROLLING OFFICE NAME AND ADDRESS: ONR Department of the Navy Arlington, Virginia 22217	12. REPORT DATE: December 1976
	NUMBER OF PAGES: 132
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acoustic-phonetic experiment facility, acoustic-phonetic recognition, acoustic-phonetic rules, artificial intelligence, ATN grammars, audio-response generation, budget management, computational linguistics, control strategies, data base, dictionary expansion, discourse model, fact retrieval, formal command language, formant tracking, grammars, HWIM, knowledge sources.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This five-volume final report is a review of the BBN speech understanding system, covering the last two years of the project from October 1974 through October 1976. The BBN speech understanding project is an effort to develop a continuous speech understanding system which uses syntactic, semantic, and pragmatic support from higher level linguistic knowledge sources to compensate for the inherent acoustic indeterminacies in continuous spoken utterances. These knowledge sources are integrated with

DD FORM 1 JAN 73 14/3 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Date Entered)

19. Key words (cont'd)

lexical retrieval, likelihood ratio, linear prediction, multi-component systems, natural language retrieval system, natural language understanding, parametric modeling, parsing, pattern recognition, phonetic labeling, phonetic segmentation, phonological rules, phonology, pragmatic grammar, pragmatics, probabilistic labeling, probabilistic lexical retrieval, prosodics, question-answering, recognition strategies, resource allocation, response generation, scoring philosophy, semantic interpretation, semantic networks, semantics, shortfall algorithm, shortfall density, signal processing, spectral matching, speech, speech analysis, speech generation, speech synthesis, speech recognition, speech understanding, SUR, syntax, synthesis-by-rule, system organization, task model, user model, word verification.

20. Abstract (cont'd)

sophisticated signal processing and acoustic-phonetic analysis of the input signal, to produce a total system for understanding continuous speech. The system contains components for signal analysis, acoustic parameter extraction, acoustic-phonetic analysis of the signal, phonological expansion of the lexicon, lexical matching and retrieval, syntactic analysis and prediction, semantic analysis and prediction, pragmatic evaluation and prediction, and inferential fact retrieval and question answering, as well as synthesized text or spoken output. Those aspects of the system covered in each volume are:

Volume I.	Introduction and Overview
Volume II.	Acoustic Front End
Volume III.	Lexicon, Lexical Retrieval and Control
Volume IV.	Syntax and Semantics
Volume V.	The Travel Budget Manager's Assistant

The BBN Speech Understanding System: Final Report

This is one of five volumes of the BBN Speech Understanding System Final Report. The Table of Contents for the entire set is given below.

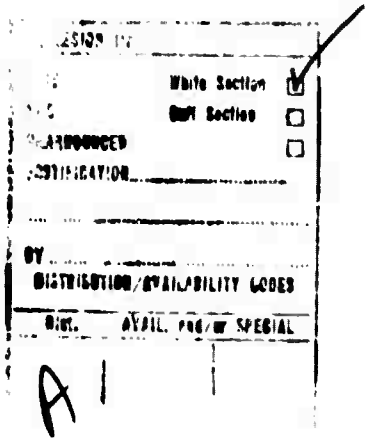
Acknowledgements

To the following people who contributed to the development of the BBN Speech Understanding System:

William A. Woods, Principal Investigator
Madeleine Bates
Geoffrey Brown
Bertram C. Bruce
Laura Gould
Craig C. Cook
Gregory Harris
Dennis H. Klatt
John W. Klovstad
John I. Makhoul
Bonnie L. Nash-Webber
Richard M. Schwartz
Jared J. Wolf
Victor W. Zue

To our secretaries - Beverly Tobiason, Kathleen Starr and Angela Beckwith - for the exceptional diligence, competence, and good humor they have shown throughout the assembly of this report.

"Had we but world enough and time..."
Andrew Marvell, To His Coy Mistress



SPEECH UNDERSTANDING SYSTEMS

Final Report

November 1974 - October 1976

ARPA Order No. 2904

Program Code No.5D30

Name of Contractor:
Bolt Beranek and Newman Inc.

Effective Date of Contract:
30 October 1974

Contract Expiration Date:
29 October 1976

Amount of Contract: $1,966,927

Contract No. N00014-75-C-0533

Principal Investigator:
William A. Woods
(617) 491-1850 x361

Scientific Officer:
Marvin Denicoff

Title:
SPEECH UNDERSTANDING SYSTEMS

Editor:
Bonnie Nash-Webber
(617) 491-1850 x227

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2904

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under Contract No. N00014-75-C-0533.

TABLE OF CONTENTS

A. PARSERS

Introduction

A special influence on the design of every parser used in HWIM has been the unique role assigned to its Syntactic Component (Vol. III, Sec. E). HWIM's control strategy requires that the Syntactic component be able to take any island (i.e., consecutive sequence of word matches) and determine if it can be parsed as an acceptable subsequence within the grammar. If so, the Syntactic component must be able to return to Control a list of words and categories that would form acceptable extensions from the island at either end. The constraints this places on the parser are that it must be able to start at any point in the input (not just the left end) and at any point in the grammar and work in either direction. It must be able to process islands that may partially or fully traverse several different levels of the grammar. Predictions of adjacent words should be based on the entire context of the island, rather than on, for example, only the words at either end. The latter constraint requires that the parser, at least for making predictions, must have a way to access the possible paths through the grammar within a given island.

1. Island Parsing

The "island parsing" mechanism to be described here supports the island driven strategies discussed in Vol. III, Sec. E. An island is a single word or contiguous string of words that covers some part of an utterance. The parser treats each island as an entity that is created either as a single word (i.e., a seed event) or by adding a single word to an existing island (i.e., as a word event). This allows Control to select the best available seed or word event for syntactic processing at each step.

The parser has four basic actions. The first, seed event processing, creates a new one-word island and a representation for every path through that word that the grammar allows. The second, word event processing, adds a new word to one end of an existing island, extending those paths that allow the new word and eliminating those paths that do not. By this mechanism, the addition of a word at one end of an island can narrow the

proposals at the other end. The third action of the parser is the end event processing. This takes place when an existing island has reached one end of the utterance. The parser then extends the paths that can reach the start state or a final state of the grammar without using additional words. If the other end of the island is open, the parser returns a set of predictions at that end, which may be more restricted than before; if that end is also complete, the parser returns a complete parse. The last type of action, island collision event processing, was added as an efficiency mechanism to allow Control to merge two islands with a one-word gap between them and a common proposal for that gap. Island collision events could easily be simulated by several word events, but a substantial savings in Control and Control-Syntax interface effort is achieved by use of this type of event. All four types of processing are discussed below.

2. Grammar Structures

a) Description of the ATN Machine

In order to better characterize the new parsing system used in HWIM, it is useful to think of its grammar as an "ATN machine", i.e., to see it procedurally rather than structurally. This will allow us to give a more concise description of the parsing algorithm and show that it will produce the same results as ATN parsers described elsewhere [Bates, 1975b; Burton, 1976; Burton and Woods, 1976; Woods, 1969, 1970].

The current HWIM grammars (see Sec. B) use only five different arc types: PUSH, POP, WRD, CAT and JUMP.

A PUSH arc essentially "consumes" a non-terminal in the grammar by causing a new transition network to be entered at a lower level. If the lower transition network can successfully accept the next segment of the input string, it is exited by a POP arc, and processing will continue in the transition network containing the PUSH arc. Each final state of a transition network will contain a POP arc that signifies successful completion of the network and indicates what structure is to be returned for use by its calling network.

WRD and CAT arcs are the only terminal consuming (i.e., word consuming) arcs in the grammar. That is, they are the only arcs that advance the "input pointer" that marks the current position in the input string. (This pointer may move during a PUSH arc, but only as a result of WRD and CAT arcs taken in the lower network.) WRD and CAT arcs differ only in the way they express the set of terminals they can consume. A WRD arc specifies its terminals explicitly and exhaustively, while a CAT arc specifies them implicitly via the syntactic-semantic category to which they must belong. Since the function of WRD and CAT arcs in the ATN system is so similar, they will be considered the same for most of the functional description of the HWIM parser.

A JUMP arc is similar to a WRD or CAT arc since it continues the given transition network. However, it does not consume a terminal in doing so and hence does not advance the input pointer.

```
<ATN> -> (<state> <state>*)

<state> -> (<state-name><arc><arc>*)

<arc> -> <W> | <B> | <C> | <J>

<W> -> (CAT <category-name> <action>* (TO <state-name>))
       (WRD <terminal "word"> <action>* (TO <state-name>))

<B> -> (PUSH <state-name> <action>* (TO <state-name>))

<C> -> (POP <form> <action>*)

<J> -> (JUMP <state-name> <action>*)

<action> -> <action1> |
            (SCOPE <scope-spec> <action1> <action1>*)

<action1> -> (VERIFY <action1>) |
             <register-setting-action> |
             <structure-building-action> |
             <testing-action>

<scope-spec> -> (<state-name> <state-name>*) |
                NIL | T
```

Fig. 1. Format of a HWIM ATN grammar

Figure 1 shows the notation we use for describing a HWIM ATN grammar; it is similar to most other ATN formalisms. In referring to an arc, the arc <u>type</u> is the first entry (WRD, CAT, etc.) and the arc <u>label</u> is the second element of the arc. There will be no discussion of the operation of a formal ATN machine here, since that has been described fully elsewhere.

The purpose of this formal description is to allow us to show the new and different features used by the HWIM parser.

b) State Relations used in HWIM Parser

In this section we will refer to the connection of two states by a particular type of arc as a relation of that name between those two states. We will use not the ATN arc types (PUSH, JUMP etc.) but, in most cases, the single letter given in Fig. 1 above. Here, W represents a terminal consuming arc (i.e., a WRD or CAT arc); J, a JUMP arc; and P, a PUSH arc. In addition, we will use B to represent the relationship between the state at the beginning of a PUSH arc and the state pushed to, and we will use C for the relationship between the state at the beginning of a POP arc and a state to which control could revert at a higher level after executing the POP (i.e., a state at the end of a PUSH arc that could push to the network containing the POP). Thus B and C represent beginning and completing a lower level of the grammar, while W, P and J represent transitions within the same level. For example, if there exists a jump arc between STATE1 and STATE2 we would say the relation STATE1 J STATE2 holds.

Since all the above relations are non-reflexive, we also need a set of relations for right to left operations: LJ for Left Jump, UB for Un-Begin, UC for Un-Complete, LW for Left terminal consuming, and LP for Left Phrase consuming. We also use the Kleene star to indicate 0 or more occurrences of a relation. For example, the relation (STATE1 J* STATE2) is true if STATE1 is the same as STATE2 or if there exists some path made up exclusively of jump arcs between STATE1 and STATE2.

We can also concatenate a set of these basic relationships to describe a path over more than one type of arc. (The only restriction is that they all have the same directionality). Thus, the relation (STATE1 J*BJ* STATE2) is true if (1) there is some path from STATE1 to STATEx that follows 0 or more JUMP arcs (STATE1 J* STATEx); (2) there is a PUSH arc in STATEx which has STATEy, the beginning of the lower level transition net, as its label (STATEx B STATEy); and (3) there is some path of 0 or more JUMP arcs from STATEy to STATE2 (STATEy J* STATE2).

We will use these relations to describe sets of states to be considered by various parts of the algorithm. The J* set of STATE1 is the set of all those states for which the relation (STATE1 J* STATEn) is true. If one is looking at a state STATE1 which has one or more PUSH arcs, and one wants to know what states at the next lower level might have an arc that could consume the next word of input, the BJ* set of STATE1 would be that set of states.

A few more comments about the relations are in order. The relations C and its inverse UB are somewhat more complicated than just following an arc, since a POP arc does not of itself indicate the next state at the higher level in the grammar. The relation (STATE1 C STATE2) is true if there is a PUSH arc which has STATEz as its TO state, and some STATEx as its label, i.e. (PUSH STATEx <action>+ (TO STATEz)), and there is some path from STATEx to STATE1 at the same level in the grammar. Since UB is the inverse of C, (STATE1 C STATE2) implies (STATE2 UB STATE1) and vice versa.

Once we think of these relations as strings, we can have other connections between them than concatenation. For example, to describe a path across a single transition network of the grammar we can say (STATE1 (W or J or P)* STATE2) or, as a regular expression, (STATE1 (W+J+P)* STATE2). Using this notation we can now name some relations that are important to the HWIM parser. These are B! and C! for left to right and UC! and UB! for right to left.

$$B! = B\ ((J + B)^*\ B)^*$$
$$C! = C\ ((J + C)^*\ C)^*\ ((J + B)^*\ B)^*$$
$$UB! = UB\ ((LJ + UB)^*\ UB)^*\ ((LJ + UC)^*\ UC)^*$$
$$UC! = UC\ ((LJ + UC)^*\ UC)^*$$

Even more useful than these relations are the relations J*B!J*, J*C!J*, LJ*UB!LJ* and LJ*UC!LJ*. These can be fairly easily described in English as the paths that will reach all possible states on other levels of the grammar that can be reached using only non-consuming arcs. The utility of these indices is fairly clear; if there is a segment configuration ending in STATE1, and a new word is to be added to the island that can be

consumed by an arc that begins at STATE2, then only if (STATE1 (J* + J*B!J* + J*C!J*) STATE2) is true, can there be a path that will connect the island to the new word.

There are two reasons why we break the relation down at the J*, B! etc. level. First, we usually treat the J*, J*B!J*, and J*C!J* cases separately, and secondly, there is an enormous space savings to storing only the B! and C! sets rather than the J*B!J* and J*C!J* sets, which are very easily computable from the individual J*, B! and C! sets. The use of these sets will be discussed in the section describing the parsing algorithm (Sec. 5).

3. Parser Data Structures

a) States of the Parsing Process

i. Configurations

In order to handle all of the requirements placed on the Syntactic component, it has proved useful to rethink the notion of a "state of the parsing process". In a standard left-to-right ATN parser, such a state is described by a configuration, containing (at least) a pointer into the input, the current state of the grammar, the current register settings, and a stack of next states and register settings for the higher levels of the computation. However, this notion of a configuration is inadequate for the HWIM parser. First, since there is no fixed "input string", there is no way to have a pointer into it. Secondly, both ends of the island must be indicated, since an island need not start at the left end. These facts also invalidate the concept of a real stack on which all higher levels of the computation are stored, since higher levels cannot be known with certainty unless the computation begins at the left end.

These problems are solved by redefining a configuration, separating it into two parts. The first part is called an island configuration (ICONFIG) and contains information internal to the island, including its word matches and an indication of whether the island reaches either end of the utterance. The second part contains a list of segment configurations (SCONFIGs), which represent the state of the computation for each level of

the grammar in which any word of the island can be consumed. A segment configuration is made up of a left boundary, a right boundary, a left state, a right state, a list of the register settings already made in this level of the parse, a list of arc actions being held until the necessary context is complete, and a list of states that have been passed through on this level. When an SCONFIG covers a complete traversal of one level of the grammar, a new constituent can be built for use by PUSH arcs at the next higher level of the grammar. The grouping of SCONFIGs under an ICONFIG alleviates the need for a complete stack or left context.

ii. Paths

Two adjacent segment configurations (referred to for brevity as "segments" in the subsequent discussion) will be said to be compatible if the left one stands in the (J*B!J* + J*C!J*) relation to the right one (i.e., the right state of the left one stands in that relation to the left state of the right one). That is, two adjacent segments are compatible if the left one either pushes or pops, perhaps indirectly, to the right one. A sequence of compatible adjacent segments will be called a path provided that there is no subsequence of three segments such that the first stands in the J*B!J* relation to the second, which in turn stands in the J*C!J* relation to the third. (Whenever such a compatible sequence of three segments exists, the middle one can be completed and consumed by a PUSH arc that incorporates it into either the left or right segment or perhaps combines all three into a single segment.) Associated with each path is an indication of which segment is at a higher level in the grammar than any other. Note that it is possible to have two paths that are identical except for which segment is chosen as the top. Marking the top segment of a path divides the remaining ones into two groups, which must then be at successively lower levels as they get further away from the top.

Paths are only necessary for making predictions off an island. Since they can be computed from an island configuration and a set of segment configurations, it is not necessary to make them permanent data structures. However the concepts of path and top segment are very helpful when considering how to connect segments together, as well as for making predictions.

b) A Middle-Out View of an ATN Grammar

In the original presentation of ATN grammars [Woods, 1970], a state was viewed as the collection of arcs leading out of it, and an arc was associated with its begin-state. While this conceptualization is adequate for left to right parsing, if we try to use it from right to left, it becomes necessary to search all states to find those arcs that can reach a given state from the left. This is clearly both conceptually awkward and computationally inefficient.

For HWIM's parser, it is more useful to think of an arc as a connection between two states that can be traversed in either direction. An arc then is associated with a left state, a right state, a type (WRD, CAT, etc.), a label, a set of context-free actions that can be done when the arc i first encountered regardless of the direction or context, and a set of context sensitive actions, which will be deferred until adequate left context is available. A state then has two associated collections of arcs, one set leading to the left and the other leading to the right. In addition, each state has several sets of other related states associated with it (discussed in Sec. 3.d) that are used to facilitate the parsing process.

To the grammar writer, the ATN remains basically a left to right machine. This means that its arc actions can be written almost as if the parser were operating only left to right. The grammar has actions on arcs where they should be executed if the entire appropriate left context were set. As in standard left-to-right grammars, in no case is it ever necessary to worry about the right context. To deal with actions that require a specific left context, the HWIM parser has a mechanism called "scoping", which will be discussed in the next section.

c) Scoping

The scoping mechanism allows a grammar that was created from a left-to-right viewpoint to be used in parsing from right to left. During normal left-to-right processing in a standard ATN parser, actions call for both accessing and changing the contents of registers. An arc action in

the grammar has a left-to-right dependency when it either requires the value of a register that is set somewhere to the left or changes the value of a register that is used somewhere to the left. In the first case, executing the action without the left context could not produce the right effect, while in the second case, executing it prematurely in a right-to-left parsing would cause the later execution of the arc action to the left to get the wrong value. Such arc actions are referred to as context sensitive. Fortunately, fewer than half of the actions in our ATN grammars turn out to have such dependencies.

By analysis of the paths through the grammar, it is possible to determine, for each context sensitive arc action, a set of states having the property that the action can be safely done if its execution in a right-to-left parse is delayed until the parse has passed through one of those states. We refer to this set of states as the scope of the action, an indication of how far left in the grammar its left-to-right dependency extends. The HWIM parser contains a mechanism for interpreting explicit scope declarations on arc actions by saving the action with its local context (the arc on which it occurred and the input being considered at the time) until its scope is satisfied (if it is not already satisfied when the action is first considered).

Two special cases of scope declarations are also included: Scope T is used to indicate an action whose execution is not to happen until the left end of the constituent being parsed is completed. This scope indication is necessary for actions on self-looping paths through the start state of some level of the network, where there is no state in the grammar that is guaranteed to be left of all dependencies. Scope NIL is used to indicate that the scope of an action is entirely local - i.e., the action is context free. This is the default interpretation for any action that is not explicitly marked for scope, but it is occasionally useful to mark it explicitly. This is true when it has an apparent dependency on another register but analysis of all preceding paths indicates that the register in question cannot be set anywhere to the left. In this case, the explicit NIL scope reminds the grammar designer that the question of scoping has been considered for this action and that the answer is NIL.

The format of a scope statement is given in Figure 1 and repeated here:

(SCOPE <scope-spec> <action 1> <action 1>*)

<scope-spec> -> (<state-name> <state-name>*) | NIL | T

The scope-spec is either a list of state names, T, or NIL. If the scope spec is a list of state names, then the action(s) can be executed when the parse begins at or has passed through any state in that list. If the scope-spec is T, the action is not to be executed until the parse has hypothesized the left end for that level of the grammar.

An example of scoping occurs in the state S/WHAT-IS-IN-BUDGET.

```
(S/WHAT-IS-IN-BUDGET
   (JUMP S/WHAT-IS-BUDGETED)
   (JUMP S/POP
       (SCOPE (S/WHAT-DO S/WHAT-HAVE)
              (VERIFY (GETR LEFT))
              (SETR SUBJ (NPBUILD)))))
```

(The location of this state in the grammar is given schematically in Figure 2.) From this state, it is possible to jump to S/POP to end the utterance only if we have a question such as "WHAT IS LEFT IN THE BUDGET." In these cases the register LEFT will have been set earlier. If we are parsing right to left, we must know how far left we must carry the parse before making this test. The JUMP arc to S/POP indicates in the scope of the VERIFY action that the parse must have passed through either S/WHAT-DO or S/WHAT-HAVE.

It is possible to construct an algorithm to compute the scopes automatically if the dependencies of the arc actions are explicitly marked, but we have not yet implemented such a facility.

d) Storing the Grammar for the HWIM Parser

As demonstrated in the LUNAR system [Woods et al., 1972], it is possible to run a parser by directly interpreting the ATN syntax as described above. In the SPARSER system [Bates, 1975b], the grammar is also used as given, augmented by a set of extra indices connecting the grammar from right to left and relating the different transition networks together in a helpful manner. In the SOPHIE system [Burton, 1976; Burton and Woods, 1976] on the other hand, the ATN grammar has been "compiled" for speed and

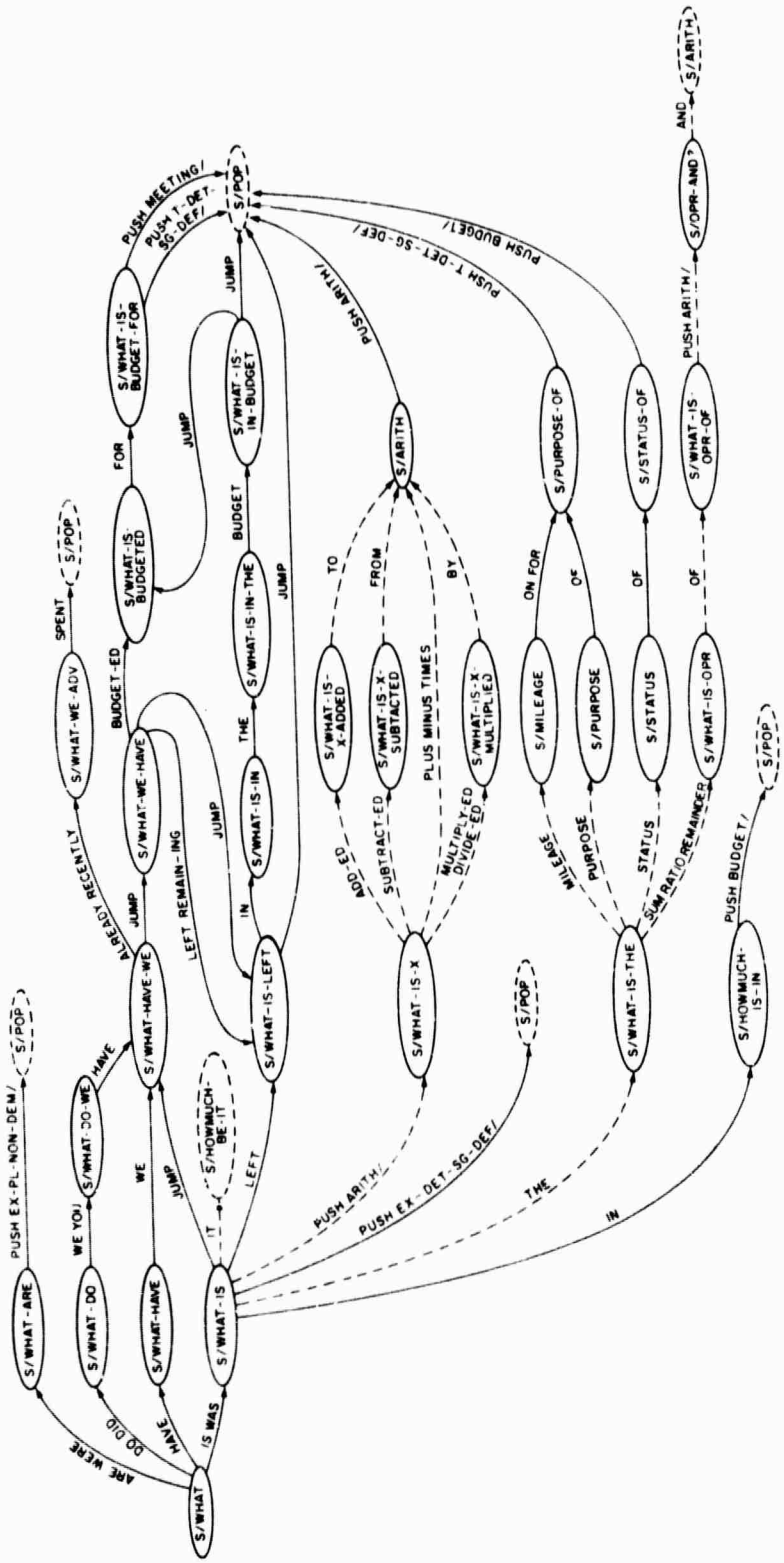


Fig. 2. A portion of the sentence-level network

efficiency into a single LISP function, though it still operates from left to right. As our desire was both to gain speed and efficiency as well as to permit bi-directional parsing. the new HWIM parser uses techniques from both SPARSER and SOPHIE. It pre-processes the grammar, smashing it into several arrays that describe the grammar in both directions. The arc arrays contain the left and right states of each arc in addition to the type, label, context free actions, and context sensitive actions associated with that arc. The state arrays contain the state name, left and right going arcs, and several precomputed state relation sets. The sets currently stored are J*, B, C, B! and C! and their right-to-left counterparts. These sets are important to the parsing algorithm in the way that the L* set used by Earley [1970] is in his description of an efficient context-free parsing algorithm. Their use is described in detail below.

4. Syntax-Control Interface

a) The Nature of the Interface

In the introduction to this section we briefly discussed the four basic functions filled by the Syntactic component in HWIM'S control strategy. Here we will describe its interface with Control.

In the current implementation of HWIM, the Syntactic and Control components inhabit separate TENEX forks, using a shared file for communication. In order to isolate the implementation specific details of the interface, Control is provided with a set of functions that have the same names and I/O characteristics as the top level functions in the Syntax fork and handle the task of calling their counterparts there.

b) High-level Parser Functions

The top level parser functions are designed to operate on only one new word at a time. That is, the parser can either consider an initial one word ("seed") island or add one new word to an existing island. This simplicity holds, even though HWIM's control strategies have become quite complex (see Vol. 3, Sec. E). The complexity at the control level lies in the selection of the next word to consider and the number of seeds in contention at the same time.

The following is a brief description of the major high-level parser functions:

SYNSEED is the function used to create an island, given a new word match and the theory number to be assigned to the new island. SYNSEED calls the function STARTISLAND, which creates all possible segments for the word match.

SYNEVENT adds a new word to an existing island, given its associated theory number, the new theory number to be assigned, the new word match to be added, and the side onto which the new word is to be added.

SYNEND takes an old theory, a new theory, and a direction and causes the old theory's island to be connected to the initial state or a final state of the grammar depending on the direction. SYNEND is necessary because islands are normally only extended through their final consuming arcs at either end, and the final non-consuming arcs necessary for completion are not done.

SYNEVENT and SYNEND can call CONNECT-LEFT and CONNECT-RIGHT to connect an arc or a state to an existing island in all possible ways. CONNECT-LEFT, CONNECT-RIGHT, and STARTISLAND make up the guts of the parser algorithm and will be discussed in Sec. 5.

The function SYNCOLLISION joins two existing islands with a new word added between them. It takes the theory numbers of the two existing islands, the new theory number to be assigned to the result, and the new word match that connects the two islands. Though it appears to violate the parser's "one-word-at-a-time" processing, it is in fact implemented by taking the larger of the two islands and adding to it both the new word and the words from the other island one at a time. SYNCOLLISION is provided to Control for convenience and efficiency.

SYNSEED, SYNEVENT, SYNEND, and SYNCOLLISION all produce the same result. They return a list consisting of:

1) a flag indicating if there were any possible parses or a complete parse for the island,
2) the score of the best path through the island,
3) the parse tree if there was a complete parse,

4) a list of categories that could be adjacent to the island on the left,
5) a list of words that could be similarly adjacent,
6) a list of categories that could be adjacent to the island on the right, and
7) a list of words that could be on the right.

5. The Principal Parser Functions

In this section we will describe the main functions in the HWIM parser, as well as give a fairly complete step by step description of each. The step by step descriptions will give some idea of how the parser accesses information and groups its computations. The functions described are all suffixed -RIGHT to indicate that they are used when adding a new word to the right of an existing island. There is a set of similar functions (suffixed -LEFT), which work on adding words to the left. The differences between the sets are small, brought on by the fact that a new level is begun at one particular state on the left (the label of the PUSH arc) but can pop from any of a number of accepting states (states with POP arcs) on the right. These differences usually involve only an extra loop to be executed a bounded number of times.

a) Starting an Island

To begin a one-word island, the parser finds all arcs in the grammar that can consume the new word match, using an index of pointers from each word and category to the arcs that consume them. It processes those arcs, creating a segment configuration (SCONFIG) for each one. These are collected into an island configuration (ICONFIG), which is then processed to find the proposals to return to Control. The proposal mechanism will be discussed Section 5.c.v.

b) Processing an Arc

The basic transition function in the parser, DOARC, takes an old SCONFIG, an adjacent arc, and, if it is a consuming arc, a word or constituent to be consumed. The result is a new SCONFIG representing the state of the computation after the new arc and input have been incorporated. DOARC is applied to a list of SCONFIGs to be processed for a given arc and direction. The result is a new list of SCONFIGs.

DOARC(SCONFIGLIST ARC WORD/CONSTIT DIRECTION):

1) For each SCONFIG do 2 through N

2) Verify that the arc is adjacent to the old SCONFIG;

3) Set up context for arc actions
   - set atom REGS to register list in SCONFIG
   - set atoms LEX, *, and FEATURES according to the input being consumed;

4) Do the context free actions on the arc;

5) For each context sensitive action on the arc:
   - if the context is not yet complete, create an undone scoped action to be added to the new SCONFIG
   - otherwise evaluate the action.

6) If the direction is to the left, see if the scope of any saved actions in the old SCONFIG has now been satisfied. If so, evaluate those actions.

7) Build up the new SCONFIG by
   - setting the new boundaries and end states
   - saving the registers and undone scoped actions
   - adding the left state of the arc to the list of states that the computation has passed through.

c) Connecting a New Word to an Island

This section describes the process of adding a new word to the right of an existing island. Since both connecting a new word on the left of an island and connecting to either end of the grammar are very similar to this, we will not describe them here.

In extending a segment at the end of an old island to meet a new word to be added, the sequence of intervening non-consuming arcs that make the connection can be all at the same level of the network (the J* case); they can change to a lower level (the J*B!J* case); or they can change levels in a way equivalent to one or more POPs followed by zero or more PUSHes (the J*C!J* case). Depending on the type of connection, and the stage of the extension, CONNECT-RIGHT keeps its partially extended segments (actually groups of functionally equivalent segments) in one of the following six queues:

1) TODOQ - contains all segments that have not changed levels at all. Initially this is all right boundary segments from the old island. Later when a segment from the old island is extended to include a new constituent, it is placed in the TODOQ.

2) C!Q - contains all segments that have changed to a higher level and may now only stay at that level or go down into a new constituent.

3) C!OPENLEFTQ - contains those segments from the C!Q that are incomplete on the left (i.e., have not been pushed for).

4) C!COMPLETEQ - contains the remainder of the C!Q, those segments that are now complete and are ready to form hole constituents.

5) B!Q - contains those segments that have changed to a lower level and therefore must not change levels again.

6) DONEQ - contains those segments whose right end state has a consuming are for the word being added to the island.

i. Description of CONNECT-RIGHT

The function CONNECT-RIGHT takes SCONFIGs that terminate at the right boundary of an existing island and joins them to a set of arcs that can consume the new word to be added to the island.

CONNECT-RIGHT(SCONFIG-GROUP-LIST TOARCS WORD):

1) Create a set of J*TOSTATES, which are the states that can J* to the left state of at least one of the consuming ares for the new word.

2) Put all groups of SCONFIGs that are at the right boundary of the old ICONFIG into the TODOQ.

3) If there is anything in the TODOQ, call EXTEND-PATHS-RIGHT with the whole TODOQ and number of "directions" set to 3. (See See. 5.c.ii for a discussion of EXTEND-PATHS-RIGHT and the notion of "direction".)

4) If there is anything in the C!Q, call SPLITC!Q to separate it.

5) If there is anything in the C!COMPLETEQ, call COMPLETE-RIGHT to create the constituents and process the PUSH arcs that can use them. Then go back to 3.

6) If there is anything in the C!OPENLEFTQ then call EXTEND-PATHS-RIGHT on the whole C!OPENLEFTQ with number of directions set to 2.

7) If there is anything in the B!Q then call EXTEND-PATHS-RIGHT on the B!Q with the number of directions (THYDIRS) set to 1.

8) Call DOARC (see Sec. 5.b) for eaeh consuming are and the appropriate group of SCONFIG groups in the DONEQ.

9) Collect the results of the DOARC calls together to form the new ICONFIG.

ii. Description of EXTEND-PATHS-RIGHT

EXTEND-PATHS-RIGHT causes jumps to states that either begin a consuming arc, push for a constituent that contains the consuming arc or pop from the eurrent eonstituent to a higher level that contains (or ean then push for) the consuming arc. EXTEND-PATHS-RIGHT begins from an SCONFIG group whose members share the same right boundary. It completes the JUMP paths on that level that are required to reach the states mentioned above.

EXTEND-PATHS-RIGHT is used in several contexts in CONNECT-RIGHT (see Sec. 5.c.i), so it must have a way of deciding which of the above states are relevant. To do this it must be supplied with a number indicating how many directions to look for a connection. If the number of directions (TRYDIRS) is 1, then EXTEND-PATHS-RIGHT only considers states on the same level as the input SCONFIGs, i.e., the states in the union of the J* sets of the right end states of the input SCONFIGs. If any of those states is the left state of a consuming arc (i.e., a TOSTATE), then all possible jump paths to reach that state are executed using DOARC (Sec. 5.b). The resulting SCONFIGs are collected into a group that is associated with the TOSTATE and placed in the DONEQ.

If the value of TRYDIRS is 2, then EXTEND-PATHS-RIGHT considers paths that lead to a lower level constituent, as well as ones on the same level as the input SCONFIGs. Each of the states in the union of the J* sets is also tested to see if it can reach a state in J*TOSTATES via a B! transition. If so, then all jump paths are completed from the input SCONFIGs to that state, and the resulting group of SCONFIGs is placed in the B!Q

If the number of directions (TRYDIRS) is 3, EXTEND-PATHS-RIGHT also considers paths that complete the segments of the input SCONFIGs and return to a higher level. These higher levels may contain one of the consuming arcs or may then push for new constituents that contain a consuming arc. Note that these paths are just the C! set described in Section 2.b. For each state that can reach a state in J*TOSTATES via a C! transition, all jump paths from the input SCONFIGs are created, and the resulting group of SCONFIGs is placed in the C!Q.

EXTEND-PATHS-RIGHT(QUEUE TRYDIRS):

1) Compute the set of FROMSTATES from the union of the J* sets of the right end states of the input SCONFIGs in the QUEUE.

2) Compute J*TOSTATES from the union of the LJ* sets of the left states of the TOARCS (the consuming arcs for the word being added in the event).

3) For each state in FROMSTATES do 4 through 6.

4) If FROMSTATE is a left state of any TOARC, then make all jump paths leading to FROMSTATE from any of the input SCONFIGs. This is done with the function GETPATHS-RIGHT, which builds up jump paths and keeps

a table of states that have been connected to the input SCONFIGs. In this way, the JUMP arcs are executed when they are needed. If they are needed again, the previous results are used.

5) If TRYDIRS is 2 or 3 and (FROMSTATE B! J*TOSTATE) is true for any state in J*TOSTATES, then GETPATHS-RIGHT is called for FROMSTATE and the resulting SCONFIGs are placed in the B!Q.

6) If TRYDIRS is 3 and (FROMSTATE C! J*TOSTATE) is true for any state in J*TOSTATES, then GETPATHS-RIGHT is executed for FROMSTATE and the resulting SCONFIGs are placed in the C!Q.

iii. Description of SPLITC!Q

The C!Q queue contains all segments that have reached an accepting state for the input SCONFIGs. However, there is a major difference between those of its segments which are now complete at both ends and those which are still incomplete at one end. In the first case, the appropriate constituent must be created and the process resumed at the higher level (COMPLETE-RIGHT). In the second case, however, all states in the C! set of the right state of the segment must still be considered for extension, since the segment has no left context.

SPLITC!Q checks the left boundary of each segment. If it is the left boundary of the island, then the segment is clearly open and is put in the C!OPENLEFTQ. If it is within the island, then SPLITC!Q picks up RSTATES, the list of right end states, from the list of segments whose right end coincides with the left end of the segment being considered. This is done by keeping an index of segments in the island by right boundary. If there is an intersection between the UB! set of the left end state of the SCONFIG in the C!Q and RSTATES, then the SCONFIG could have been pushed for from one of those segments, so the SCONFIG is put in the C!COMPLETEQ. If there is an intersection between the UC! set of left end states of the SCONFIG and RSTATES, then it is possible for the SCONFIG to belong to a path in which it is the highest level SCONFIG. It is therefore put in the C!OPENLEFTQ. It is possible for an SCONFIG to be in both queues as a result of different possible paths. If an SCONFIG has been put in neither queue though, a processing error has occurred, in that the island did not contain a proper set of segments.

SPLITCIQ:

1) For every SCONFIG in CIQ do 2 through 5.

2) If the left boundary of SCONFIG (LBDRY) is the same as the left boundary of the ICONFIG being processed, then put the SCONFIG in the CIOPENLEFTQ. Skip 3 thru 5.

3) Collect a set of RSTATES from the union of the right end states of the segments that have a right boundary equal to LBDRY.

4) If the intersection of RSTATES and the UBI set of LSTATE (the left end state of SCONFIG) is non-empty, then put SCONFIG in the CICOMPLETEQ.

5) If the intersection of RSTATES and the UCI set of LSTATE is non-empty, then put the SCONFIG in the CIOPENLEFTQ.

iv. Description of COMPLETE-RIGHT

In the course of adding a new word to the right of an existing island, COMPLETE-RIGHT is called if a segment that is complete at its left end encounters a final state on its right. COMPLETE-RIGHT creates the new constituent to be popped, joining it to the island in one of two ways. It may add it to an existing segment at the next higher level, or if such a segment does not exist, it will create a new intermediate segment for each arc that can use the completed constituent and is compatible with some SCONFIG of the left.

Completing the segment and creating the new constituent COMPLETE-RIGHT causes all undone scoped actions to be executed and creates the new constituent from the label of the POP arc. COMPLETE-RIGHT then looks to see if any existing segment immediately to the left of the new constituent could have pushed for it. If so, its PUSH arc is executed to include the new constituent. COMPLETE-RIGHT also looks to see if any segment to the left of the new constituent can push for it indirectly and still reach the consuming arc for the event. For each of these cases, COMPLETE-RIGHT must create a new segment, extend it from its beginning state to the state that pushes for the new constituent, and execute that push arc to create a new segment that is complete on the left and that includes the new constituent on the right.

After a segment has been extended to include the new constituent, it is placed back in the TODOQ because it may need to be further extended. The CONNECT-RIGHT function will deal with this correctly since the new segment behaves very much like the segments that were in the old island.

COMPLETE-RIGHT:

1) Select from the C!COMPLETEQ, all those SCONFIGs that cover the smallest section of the utterance (i.e., those with the right-most left boundary).

2) For each SCONFIG selected, do all remaining undone scoped actions.

3) For each SCONFIG, do the POP arc action (saved as a scoped action with scope POP) to create the constituent. Group the constituents by start state. These groups will represent ambiguous parses of a constituent with the same input.

4) Find all SCONFIGs in the old island that end at the same place the new constituents start. Name these the PUSH-SCONFIGs. Name the union of the right states of the PUSH-SCONFIGs the PUSHSTATES.

5) For each constituent group formed in 3, do 6 through 7.

6) Find all arcs that can PUSH for the new constituent (i.e., the label of the PUSH arc is the same as the start state of the constituent), and satisfy (RSTATE is a member of TOSTATE) or (RSTATE J*B!J* TOSTATE) where RSTATE is the right state of the PUSH arc and TOSTATE is any state in TOSTATES described under CONNECT-RIGHT.

7) For each such PUSH arc do 8 through 9.

8) For all PUSH-SCONFIGs whose right state is the same as the left state of the PUSH arc, extend the segment by doing the PUSH arc for each constituent in the group. Put each resulting SCONFIG on the TODOQ.

9) For each PUSH-SCONFIG whose right state can B!J* to the left state of the PUSH arc (LSTATE), do 10 through 11.

10) Find all states STATE1 such that (RSTATE of PUSH-SCONFIG B! STATE1) and (STATE1 J* LSTATE).

11) Create a new segment for each path from each STATE1 to LSTATE. Extend each of those segments by doing the PUSH arc for each of the constituents in the group. Put the resulting SCONFIGs on the TODOQ.

v. Description of PREDICT-RIGHT

PREDICT-RIGHT creates a list of all terminal consuming arcs that can be reached by any path of non-consuming arcs from the right end of an island. The word and category predictions made by the Syntactic component are collected from this list and a corresponding one for the left end of the island.

It is not sufficient for PREDICT-RIGHT to list all arcs that can be reached from states that can end the island; these predictions must be restricted by segment configurations within the island that are at a higher level than segments on the right. To make the list of predicted arcs, PREDICT-RIGHT groups the SCONFIGs by end states and boundaries so that ambiguous parses of one level are together. For each SCONFIG group at the right boundary, PREDICT-RIGHT checks to see if it can reach the left end of

the island without passing through a higher level segment. If so, then all consuming arcs in all states reachable by (J* + J*B!J* + J*C!J*) are predicted. If not, PREDICT-RIGHT divides the predictions into two cases. First, all terminal consuming arcs in states reachable from the SCONFIG by (J* + J*B!J*) are predicted. Secondly, for each SCONFIG that can push for the segment at the right boundary, PREDICT-RIGHT repeats the prediction process on the higher level SCONFIG as if its right end state were the right state of the PUSH arc that pushes for the segment at the right boundary. As a result of this process, all predictions on the right compatible with possible paths across the island have been generated.

6. Conclusion

In this section, we have attempted to give a detailed exposition of HWIM's middle-out, bi-directional ATN parsing algorithm in sufficient detail that it could be implemented by other experimenters and its properties verified either theoretically or empirically. We could have gone into more meticulous detail and attempted to give a proof of its correctness, but space and time constraints on this report have made that impossible. We believe that the exposition is essentially correct and hope that it is useful to others.

B. GRAMMARS

1. Historical Perspective

The very first grammar to be used as part of the BBN speech system was a subset of the LUNAR grammar [Woods et al., 1972]. Although it initially comprised only 11 states, it was gradually enlarged until it reached 83 states. It was named SPEECHGRAMMAR.

In adapting ATN grammars for use in the uncertain world of speech processing, a number of changes were made to the ATN grammar formalism. First, the test portion of each arc, which was formerly a single LISP form, was broken into two tests, one comprising the context-free checks that could be made on the current word and its syntactic features and the other, the context-sensitive tests, which required information stored in registers.

Another change to the grammar formalism was the removal of SENDR and HOLD actions. This did not in any way reduce the power of a grammar and moreover facilitated more efficient local parsing. As a consequence, a relative clause could be parsed and stored in a well-formed-substring table (WFST) as a complete constituent in itself, rather than only as part of some noun phrase. In such a relative clause constituent, a dummy node such as **NP** would be placed in the parse tree to hold a place for the head. This would then be replaced with the appropriate structure on the PUSH arc looking for a relative clause. This method eliminated the need to reparse the relative clause each time there were two or more alternative heads posed for the construction.

The most important characteristics of the original grammar were that it processed words by their usual parts of speech and that it constructed ordinary syntactic parse trees for a wide variety of complex syntactic constructions such as reduced and unreduced relative clauses, sentential complements, simple conjunctions, possessives, nominal compounds, and adverbial modifiers.

2. The First Speech Parser

As SPEECHGRAMMAR was being developed, a parser was also under construction that was quite different in organization and operation from that of the LUNAR system. The main features of this parser were the following:

1) It was designed to start parsing anywhere in the input stream and to parse despite the lack of certainty as to the exact nature of the words at each point in the input.

2) Complete constituents, when found, were stored in a well-formed-substring table along with their features, boundaries, and a semantic evaluation of their meaningfulness so that they might be used by any other parse path which needed a constituent of that type at the same place without reparsing.

3) As partial parse paths were built up, their pieces were also stored in tables so that any other parse that could use them did not need to reparse common sections of input.

4) Using the grammar, the parser could make predictions about the words or syntax classes that could be used to extend a sequence of words in a theory either to the right or to the left. If a gap between words was small enough to contain just one word, the parser could predict just the class or classes of words to fill the gap.

5) The control structure of the parser could be modified fairly easily to allow experimentation with various combinations of backup, sequential, and parallel search. It used a combination of depth-first and breadth-first techniques, usually following a single path but splitting into parallel paths when desirable.

6) Care was taken to allow the parser to interact frequently and easily with other components of the system (notably Semantics) in order to receive guidance and to verify completed constituents.

7) Although at any given moment the parser was concerned with only one theory, its data base contained all the information it discovered in processing previous theories, thus allowing considerable sharing of information without duplicating effort. This organization allowed for the occurrence of some event (such as the completion of a constituent) to alert the Control component to the fact that certain previously processed theories might be affected by the event and should be queued for further processing.

For a complete description of this parser, see [Bates, 1975a; Bates, 1975b].

This parser depended on semantic and pragmatic pre-selection of sets of possible words from the word lattice to cut down the combinatorial explosion of possible partial parse paths. It was able to use semantic guidance during parsing using functions that could be called on arcs of the grammar, and it ordered the parse paths so that the best, i.e., most likely according to some predetermined metric, could be tried first using a basically top down algorithm with backup to less likely alternatives. By

using a well-formed-substring table in which constituents were placed together with a score of semantic well-formedness, the parser was able to use semantic information to help order alternative possibilities.

3. Text Grammars vs. Speech Grammars

In parsing text, one may usually assume that the input is grammatical and the words are known precisely and correctly. In a speech environment, however, one must assume that even if a grammatical sentence is uttered, there will be enough error in the acoustic and lexical processing to produce high scoring but incorrect (and frequently ungrammatical) sequences of words. Thus, it is not enough that the grammar recognize correct sentences -- it must also be able to identify ungrammatical sequences as soon as possible in order to reject them. This means that given any sequence of words within a sentence (not necessarily starting at the beginning) the grammar should predict only those adjacent words that would make a sequence that could actually be part of a grammatical sentence. A great deal of time was spent on grammars used by the speech system to tighten them sufficiently, a very difficult process for large complex grammars.

4. SMALLGRAM

As we gained experience with the speech system, several problems became apparent. Because of the way the parser was constructed, a path through a constituent had to be completed before the actions on the arcs of the path could be executed. Thus it was difficult to apply syntactic and semantic constraints as early as one would have liked. For example, it was impossible to eliminate a sequence like "a trip for my recent Chicago trip" until it had been completely parsed. Similarly, long strings of incompatible adjectives ("prst left available high expensive") would be constructed before a semantic check could be made.

To eliminate this problem, two actions were taken. One was to begin work on a parser that could take more context into account as it parsed islands. This parser has been described in detail in Section A. The other was to write a new grammar for the system which used "semantic" categories

on its arcs as well as the traditional syntactic ones. The dictionary was modified to mark entries with such new semantic categories as CONVEYANCE (bus, train), DURATION (year, week), NUM-ADJ (nearly, almost, under), SEASON (fall, winter), SPONSOR (IFIP, ACL), and TRIP-ADJ (remaining, European, taken). All syntactic parts of speech were retained.

The grammar written to take advantage of this additional information was initially called SMALLGRAM in recognition of the fact that, intuitively, it was smaller in scope that SPEECHGRAMMAR. It contained a large proportion of WRD arcs and used both syntactic and semantic categories on CAT arcs. Most importantly, the set of constituents pushed for was expanded to include such "semantic constituents" as MEETING, TRIP, PERSON, etc.

To illustrate the qualitative difference between these two approaches, Figure 1 shows a schematic for simple noun phrases as were used in SPEECHGRAMMAR. Figure 2 shows a corresponding grammar fragment such as that used in SMALLGRAM.

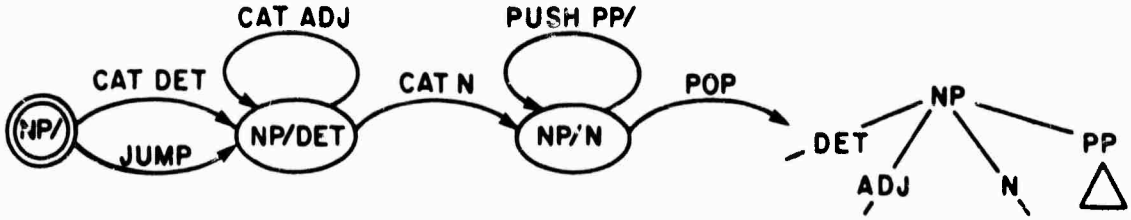


Fig. 1. A portion of a "syntactic" ATN grammar.

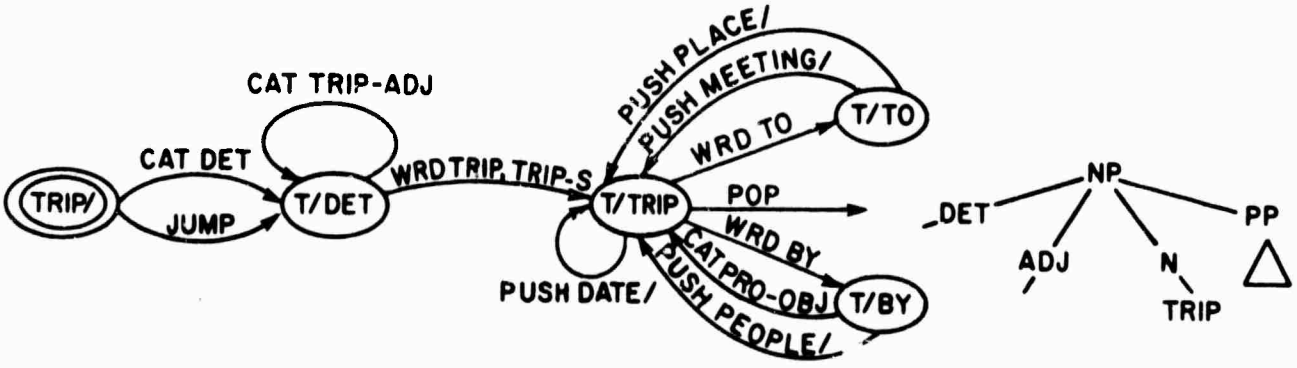


Fig. 2. A portion of a "semantic" ATN grammar.

While the structures produced by SMALLGRAM were still syntactic parse trees, they were guaranteed to be interpretable. In addition, sentences that were syntactically ambiguous in SPEECHGRAMMAR (e.g., "The money was spent by John" and "The money was spent by July") were unambiguous in SMALLGRAM because the semantic information needed to determine correct scope and modifier placement was built in.

There are several drawbacks, however, to a SMALLGRAM-type semantic grammar. One is that the grammar grows very rapidly as it is extended to handle a larger class of utterances. In fact, to handle the same set of meaningful utterances as an equivalent syntactic grammar a much larger semantic grammar is required. In addition, the task-specific nature of a semantic grammar means that it must be written anew for each different application domain. It has, however, a significant advantage for speech understanding: it provides a uniform framework for applying constraints that take into account semantic and pragmatic as well as syntactic information.

5. BIGGRAM

While SMALLGRAM used both semantic and syntactic information in determining and parsing an utterance, its final output was only a syntactic tree. Thus, information was being thrown away which had to be recovered later by semantic interpretation rules. We decided to eliminate this middle step by modifying the grammar to produce semantic interpretations directly. This not only speeded up the system but reduced its size by an entire TENEX fork.

The resulting grammar builds interpretations by accumulating in registers the semantic head, quantifier, and links of the nodes being described in the sentence. For example, the sentence

"I will go to Chicago for the ASA meeting."

yields the interpretation

```
(FOR: THE X / (FINDQ: LOCATION (CITY CHICAGO)) : T;
  (FOR: THE Y / (FINDQ: DB/MEETING (SPONSOR ASA)) : T;
    (BUILDQ: DB/TRIP  (DESTINATION X)
                      (TRAVELER SPEAKER)
                      (TO/ATTEND Y)
                      (TIME (AFTER NOW))))
```

This interpretation is built up in the following way. The PUSH arc that looks for a constituent describing a person at the start of the sentence will transform the pronoun "I" into the link-node pair (TRAVELER SPEAKER) and return this as the interpretation of that constituent. The word "will" adds (TIME (AFTER NOW)) to the list of link-node pairs being accumulated. (The grammar does not accept constructions like "will have gone," so "will" can currently always be interpreted as marking a future event.) The word "go" sets the semantic head to DB/TRIP. The constituent "to Chicago" parses with the interpretation

(DESTINATION (! THE X LOCATION (CITY CHICAGO)))).

(The ! indicates that a FOR: expression will have to be built as part of the interpretation.) Similarly, "the ASA meeting" produces

(TO/ATTEND (! THE Y DB/MEETING ((SPONSOR ASA)))).

The top level of the grammar has thus accumulated the link-node pairs

(DESTINATION (!---))
(TIME ---)
(TRAVELER SPEAKER)
(TO/ATTEND (!---)))

with the semantic head DB/TRIP. The appropriate action (in this case a BUILDQ:) is created, and the necessary quantificational expressions are expanded around it.

Currently, each level of the grammar produces the regular syntactic parse tree and pops the semantic interpretations that it has built in parallel as a feature. As soon as the grammar has been thoroughly checked out, the syntactic registers and parse trees will be eliminated.

A new action named VERIFY was also introduced into the grammar. Its effect is to abort an arc if its argument evaluates to NIL, which allowed the test component of every arc to be removed. Now the third and all subsequent elements of any arc are actions, any of which may be a VERIFY.

Because the grammar was also being enlarged to accommodate a larger vocabulary at the same time these changes were being made, it was renamed BIGGRAM. (Two of the networks in BIGGRAM are pictured in Figures 3 and 4.

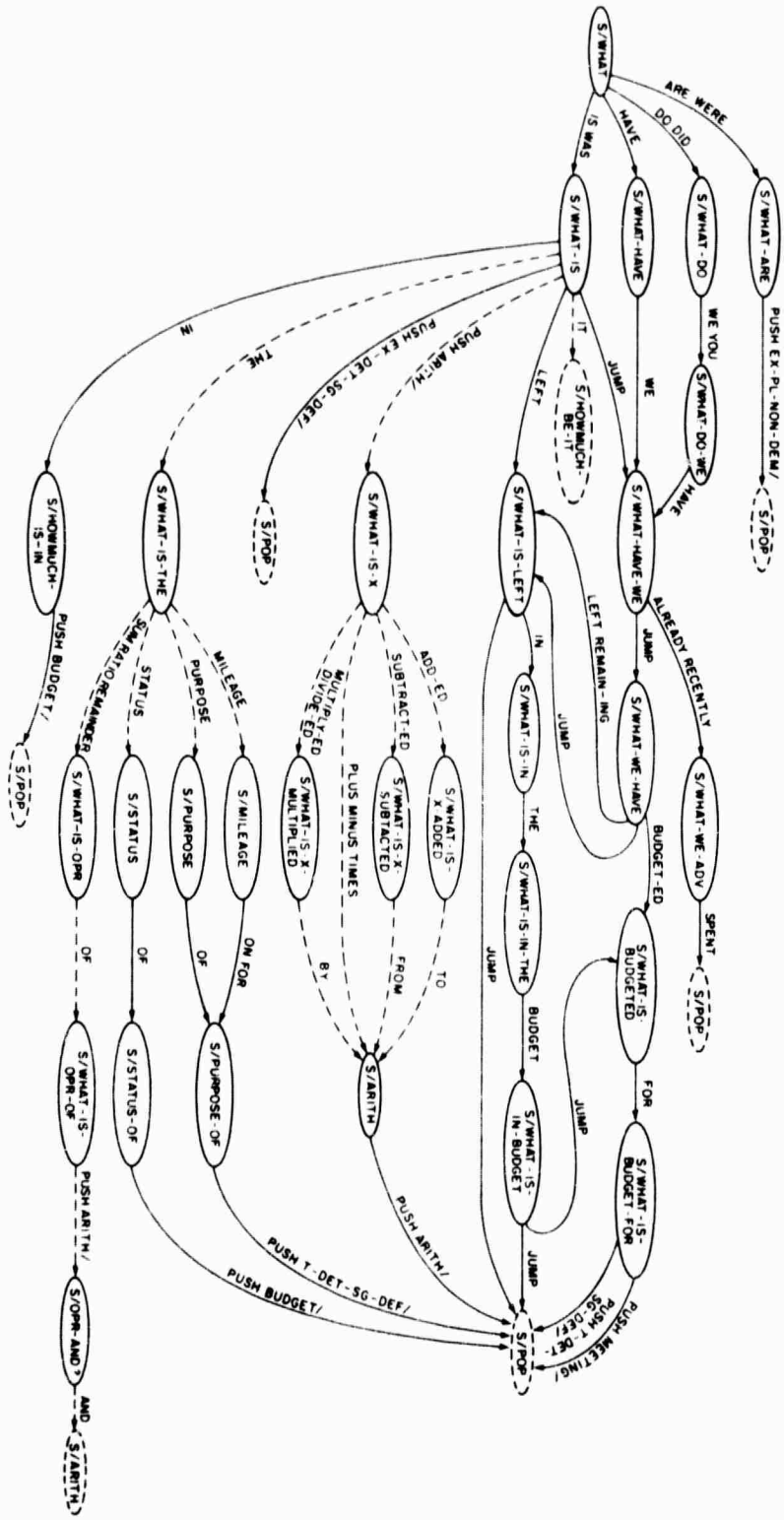


Fig. 3. A portion of the sentence-level network.

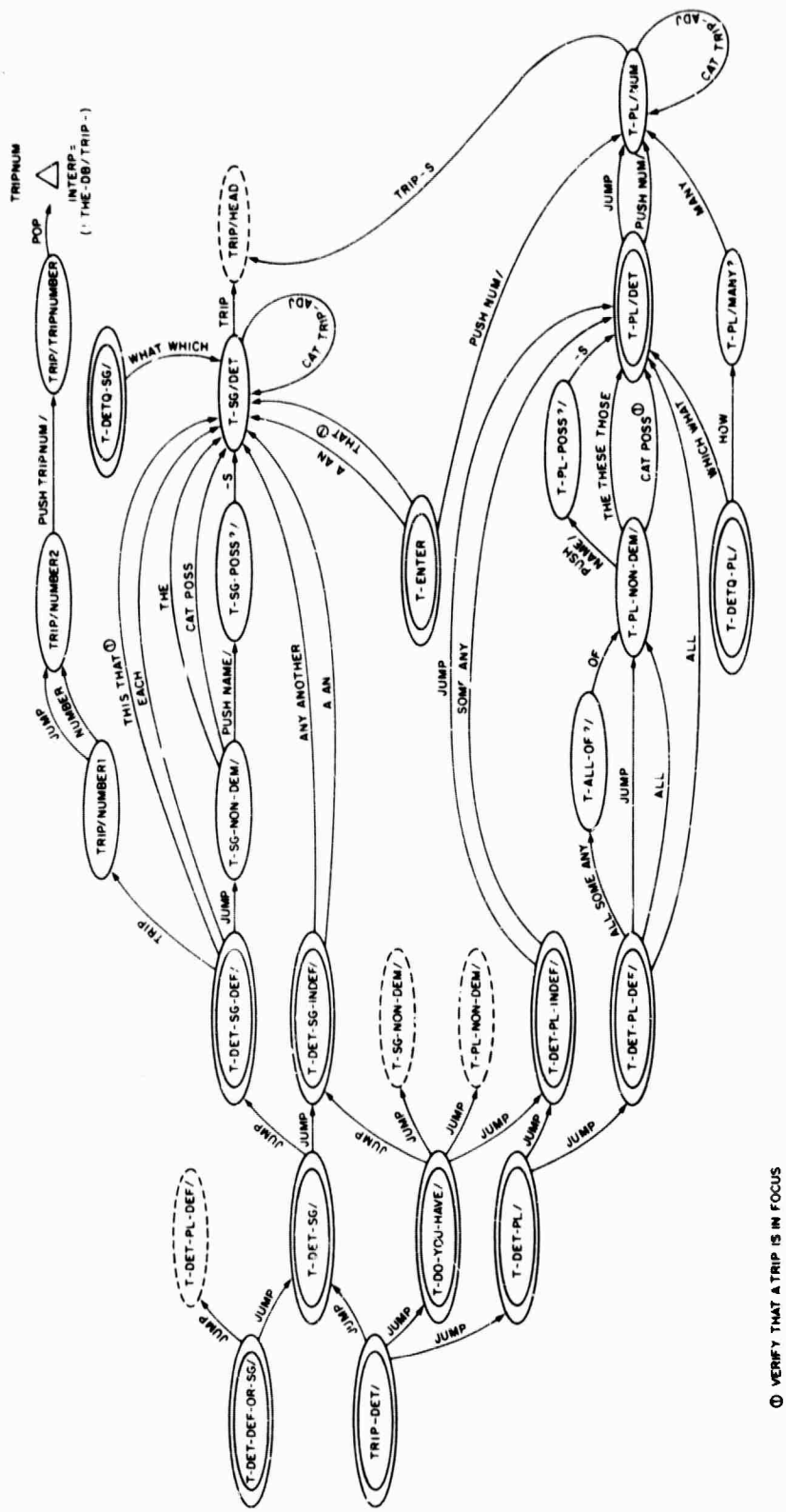


Fig. 4. The TRIP/ network.

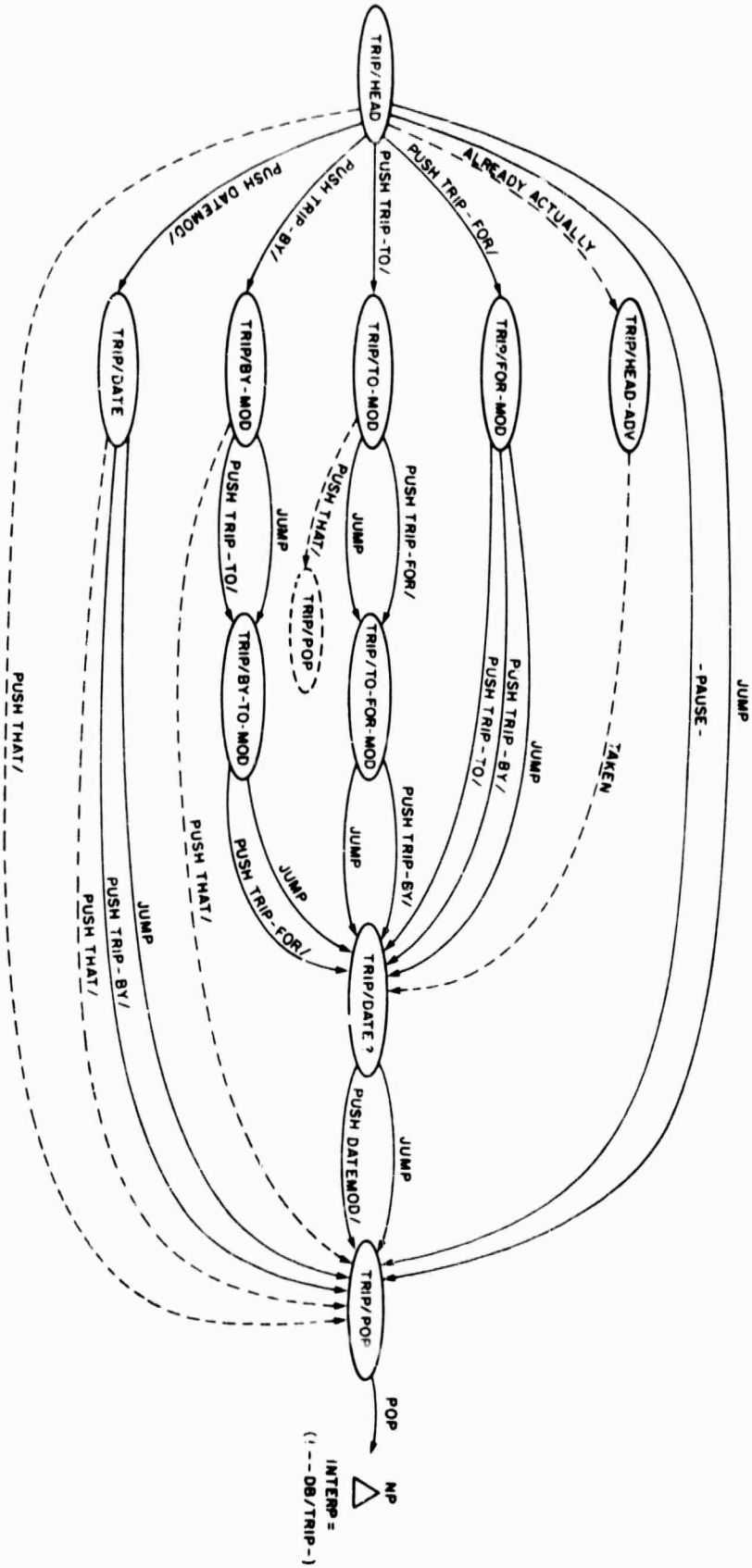

TRIP/HEAD
PUSH DATEMOD/
PUSH TRIP-BY/
PUSH TRIP-TO/
PUSH TRIP-FOR/
ALREADY ACTUALLY
TRIP/DATE
TRIP/BY-MOD
TRIP/TO-MOD
TRIP/FOR-MOD
TRIP/HEAD-ADV
PUSH TRIP-TO/
JUMP
PUSH THAT/
TRIP/POP
PUSH TRIP-FOR/
PUSH TRIP-BY/
TRIP/BY-TO-MOD
TRIP/TO-FOR-MOD
PUSH THAT/
JUMP
PUSH TRIP-FOR/
PUSH TRIP-BY/
PUSH TRIP-TO/
TAKEN
- PAUSE -
PUSH THAT/
TRIP/DATE ?
PUSH DATEMOD/
TRIP/POP
POP
NP
INTERP = ('--DB/TRIP-)

Fig. 4. (cont'd.)

The first recognizes the latter part of questions that have begun with the word "what", and the second recognizes noun phrases denoting trips. Arcs that are not also part of MIDGRAM (see Sec. 6 below) have been indicated with dashed lines.)

The interpretations that the parser builds are forms in a command language that is described fully in Volume 5 of this report. Highlights will be given here in order to discuss the way in which the grammar builds the interpretations. The most common form created by the parser is

```
(FOR: quant var / (FINDQ: nodetype (link value)...)
: test ; action).
```

This is very similar to semantic interpretations produced by the LUNAR system [Woods et al., 1972].

The function FINDQ: is a generating function that enumerates nodes of the named type having the appropriate links. For example, a node representing a trip that was already taken could be found by (FINDQ: DB/TRIP (MODALITY TAKEN)). The quantifier, quant, controls the calling of FINDQ:. Each value returned by FINDQ: is bound to the indicated variable, var, which may then be used in the test (a further restriction on the value produced by FINDQ:) and the action.

Interpretations may be nested. For example, "What is the cost of my trip?" has the interpretation

```
(FOR: THE A0127 / (FINDQ: DB/TRIP (TRAVELER SPEAKER)) : T ;
  (FOR THE A0128 / (FINDQ: DB/COST (COST/OF A0127)) : T ;
    (OUTPUT: A0128)))
```

Another common interpretation form is TEST: which prints "yes" or "no" depending on the Boolean value of its argument. So "Are we over our budget?" has the interpretation

```
(FOR: THE A0106 / (FINDQ: DB/BUDGET (BUDGET/FOR SPEAKER)) : T ;
(TEST: (OVERBUDGET? A0106)))
```

In order to modify the data base, there is a function BUILDQ: which is like FINDQ: except that it creates a node rather than finds one in the network. For example, "Enter two trips" would parse into

```
(FOR: 2 A0138 / (BUILDQ: DB/TRIP) : T ; T)
```

The only other unusual interpretation form involves the functions COUNT: and SETOF: which are used to answer "how many" questions. For example, "How many trips were taken?" would produce the interpretation

```
(FOR: THE A0079 / (SETOF: (FINDQ DB/TRIP (TIME (BEFORE NOW))))
  : T ; (OUTPUT: (COUNT: A0079)))
```

For most sentences, building semantic interpretations in registers on arcs of the grammar is extremely straightforward. For each semantic entity like TRIP and BUDGET, a FOR: form will eventually be developed, so the information that must be accumulated is a quantifier, a head (which will become the node type in the FINDQ: or BUILDQ:), a set of semantic link-value pairs, and, perhaps, a test and an indication of which function (if not FINDQ:) is to be used in the body of the form. The register names SEMQUANT, SEMHEAD, SEMLINKS, SEMTEST, and SEMFORM are used to hold these values. The register SEMVAR is occasionally used to hold the variable, which is created via the LISP function GENSYM. When a constituent has been parsed, a pseudo-interpretation is constructed on the POP arc. This is done since not all the information may be available at the given constituent level. For example, in the sentence, "How many trips did John take in July?", the initial PUSH for a trip will find "how many trips" and the information from the rest of the sentence (traveler and when) will have to be added at the S/ level. A pseudo-interpretation is a list of the form

```
(! quant var head semlinks test function)
```

The ! identifies this as a pseudo-interpretation. The quantifier, variable, and head are required, but the links, test, and function may be empty. If the function is empty, FINDQ: is inserted in the final form, otherwise, the indicated function (generally, BUILDQ:) is used.

In cases where the constituent may be augmented at a higher level, the PUSH arc that receives this pseudo-interpretation may take it apart and put the values back in registers at the higher level so that they may be added to or changed. If, instead, the constituent modifies something else, the pseudo-interpretation will be put into a new register (most likely the

SEMLINKS register of the higher level). For example, the fragment "Chicago" would be parsed by the CITY/ network into (! THE A0021 DB/LOCATION ((CITY CHICAGO))). On a PUSH arc in the TRIP/ network ("a trip to Chicago"), this would be added to the SEMLINKS register under the link DESTINATION so that the eventual interpretation of the TRIP/ network would be

```
(! A0022 DB/TRIP ((DESTINATION (! THE A0021 DB/LOCATION ((CITY
CHICAGO))))))
```

This in turn may be embedded in yet another constituent.

Eventually, the top level of the grammar will create such a pseudo-interpretation together with an action (usually deriving from the main verb) in the ACTION register. At the final POP a function called INTERPBUILD is called that takes this form and action and produces the corresponding real interpretation in the command language described above. It turns the pseudo-interpretation inside out so that every embedded (! ...) form becomes an enclosing (FOR: ...) form in which the original pseudo-interpretation is replaced by the variable it contains. For example, "Show me Bill's trip to Washington" produces the pseudo-interpretation

```
(! THE A0058 DB/TRIP ((TRAVELER (! THE A0057 DB/PERSON
                                   ((FIRSTNAME BILL))))
                      (DESTINATION (! THE A0056 LOCATION
                                      ((AMBIG WASHINGTON)))))
```

which (together with the action (OUTPUT: A0058)) is transformed into

```
(FOR: THE A0056 / (FINDQ: LOCATION (AMBIG WASHINGTON))
  : T ; (FOR THE A0057 / (FINDQ: PERSON (FIRSTNAME BILL))
    : T ; (FOR THE A0058 / (FINDQ : DB/TRIP (DESTINATION A0056)
     (TRAVELER A0057))
      : T ; (OUTPUT: A0058))))
```

Building semantic interpretations in this manner is almost, but not quite, as easy as it seems. In some cases the correct quantifier for a constituent cannot be determined until more global information is available. For example, "a trip" should be interpreted generically in a declarative sentence ("A trip to Chicago costs ..."), as "one trip" in an imperative sentence with a "constructive" verb ("Schedule a trip...") and as "some trip" in a yes/no question ("Is there a trip...?"). This is accomplished by having a pseudo-quantifier, SOME/ONE, which is later

modified to either SOME or ONE. (Generic is interpreted as ONE.) Similarly, a pseudo-quantifier HOWMANY is allowed in the construction of pseudo-interpretations of noun phrases which is later transformed into the appropriate structure using COUNT: and SE. F: (see above).

here are a number of advantages to this scheme. In fact, if the gramm ad been written from scratch instead of being extended from SMALLGRA.. which built both syntactic parse trees and semantic interpretations, it would have been much smaller. A new and specialized function was written to add link-value pairs to the SEMLINKS register, rather than always using ADDL. This function, INCORP, incorporates the new pair after checking to see that information was not being duplicated in an invalid way. For example, the sentence "John's trip that Bill took" would be extremely difficulty to eliminate on syntactic grounds alone, but semantically it would involve two different (TRAVELER x) pairs on the SEMLINKS list. INCORP checks to see if the pair it is adding is already represented on the list and aborts the arc if it is incompatible. In the case of time expressions, however, INCORP must perform a more complex check, because while time expressions may be picked up at several places in the sentence, only the most specific should be kept. Thus, "How many trips did John take?" should contain an indication that the trips were in the past, but "How many trips did John take in July?" should contain (TIME LAST (MONTH JULY)), not (TIME (BEFORE NOW)).

Information about dialogue dependent matters resides in the TRIP fork, and there are several places where the grammar calls it directly in order to make pragmatic checks on a particular parse path. For example, the phrase "that meeting" should only be allowed if the sentence is part of a dialogue in which a meeting has been mentioned. The action (VERIFY (ASKFORK TRIP (FOR: SOME X / (FINDQ: DB/MEETING) ; T : (INFOCUS? X)))) is used on an arc to make this inquiry. If the answer is false, the arc is aborted. (In single sentence mode, the function INFOCUS? is made to return T). Note that this check can be made as soon as the word "that" is considered; it is not necessary to process as far as "meeting", since it is being picked up in the MEETING/ network.

Similarly, cities and states, cities and countries, and firstname/lastname pairs must denote entities known to the system. This is checked by the function GOODPLACE (or GOODNAME) which calls the TRIP fork to validate the combinations proposed by the grammar.

This sort of checking must be done with care. For example, "two trips to Baghdad" should not be verified pragmatically until a larger context is established, since the phrase by itself cannot indicate whether they are trips that the system should know about (e.g., "Cancel two trips to Baghdad."), may not know about (e.g., "Are there two trips to Baghdad in the budget?") or should not know about (e.g., "Add two trips to Baghdad.").

6. MIDGRAM

To experiment with grammars of different sizes, it was decided to create a grammar called MIDGRAM, which is a subset of BIGGRAM. To facilitate this operation, a pseudo-action NOTIN was created. NOTIN appears, if at all, as the third element of an arc and has as arguments the grammars that do not include this arc. (It was envisioned that someday a TINYGRAM would exist also.) A procedure was written to extract a grammar X from BIGGRAM by eliminating from BIGGRAM all arcs which contained a (NOTIN ...X...) action. If all arcs were removed from a state, the state itself was removed from the resulting grammar. Appendix 1 gives a listing of MIDGRAM. Appendix 2 shows some sample parses using BIGGRAM. The following chart compares several of the grammars used by the speech system.

	BIGGRAM	MIDGRAM	SMALLGRAM	SPEECHGRAMMAR
#states	448	337	399	83
#arcs	881	648	773	202
#pbdrys	77	62	0	0
#actions[1]	2280	1649	1467	386
#CAT arcs	62	55	60	40
#WRD arcs	439	306	366	52
#words in TRAVELDICT used[2]	-	409	422	-
#words in BIGDICT used[3]	1097	-	-	-

(blanks indicate not applicable or measurements not made.)

[1] excluding NOTIN and PBDRY pseudoactions.

[2] using the expanded version of TRAVELDICT which contains 667 words.

[3] using the expanded version of BIGDICT which contains 1363 words.

7. Functions Used By The Grammar

(ARTAGREE det wrd) predicate which checks phonological compatibility of an indefinite article with the word following it. Will screen out "a item", etc.

(ASKFORK fork form) calls the indicated LISP fork to evaluate form.

(CATCHECK word category) predicate to test whether word has the part of speech category.

(CATF word category feature) returns the value of feature for word as the part of speech category.

(CONJOINABLE args) takes a list of possibly conjoined phrases and verifies that no two are the same, an that if the first one is a pronoun, it isn't "I" or "me" and all the others must be pronouns; rejects combinations like "them and them".

(GOODNAME first last) calls TRIP fork to verify that first and last names go together.

(GOODPLACE city state) calls TRIP fork to verify that city and state (which may actually be a country) go together.

(INCORP reg pair) adds pair to the contents of the named register reg only if the links don't conflict (see discussion above).

(INTERBUILD form) when form is (! ...) produces an expression in command language.

(MERGE reg newlist) calls INCORP to add each element (pair) of newlist to the register reg.

(NOTIN grammar) a pseudo-action used to remove arcs to get a smaller grammar.

(NUMBOF form) returns the number represented by form.

(PBDRY) checks for a prosodic boundary preceding the current word. See Section IV. C.

(PNCHECK nounphrase code) checks person-number agreement between the nounphrase and the code that was obtained from the verb; e.g., "travel" has code X 3SG because it can go with any form except 3rd person singular.

(RIGHTENDP) returns T if the end of the utterance has been reached; returns NIL otherwise.

(SCOPE [state1 ... staten] form) requires that one of the named states has been encountered on the path to the left before evaluating form. This is useful when parsing right to left when form uses registers that are set in a state in the list.

(SEMPRO pronoun) returns the form of the pronoun to be used in building semantic interpretations, e.g., I, ME -> SPEAKER; HE -> (! THAT PERSON ((GENDER MALE))); etc.

(SPREADR [reg1 ... regn] form) takes the value of form which should be a list of elements, and sets the registers to those elements in pairwise fashion. It calls INCORP rather than SETR if the register name is SEMLINKS.

(TIMESENSE tense form) tests that tense does not conflict with a time word (e.g., next, since) in form.

(TRIP-ADJ-CK) checks compatibility of the modifiers in a trip phrase to screen out combinations like "previous taken" or "European domestic".

(VERIFY form) evaluates form and causes the arc to be aborted if the value is NIL.

(WHOLEPROJ? form) returns T if form is a complete project name.

C. PROSODICS

One source of knowledge available to speech understanding systems is the interpretation of the suprasegmental information contained in the fundamental frequency contour of a sentence. Lea's earlier research [Lea, 1972, 1973; Lea et al., 1973] showed that a decrease in fundamental frequency usually occurs at the end of each major syntactic constituent, with an increase usually near the beginning of the next one. He proposed an algorithm for "detecting" syntactic boundaries by recognizing this fall-rise pattern in F_0. Phonetic effects (especially unvoiced consonants) can also cause such a fall-rise pattern, but the effect is generally somewhat smaller, so they can be screened out by requiring that an F_0 decrease exceed a "fall threshold" and that an F_0 increase exceed a "rise threshold" in order for a boundary to be recognized.

Our earlier work with Lea's algorithm [Bates and Wolf, 1975] pointed out problems with:

1. The accuracy of the F_0 data produced by our Signal Processing component. (This has since been largely remedied by a newer F extraction algorithm, described in Vol. I, Sec. A.)
2. The interpretation of the boundary locations detected by the algorithm.
   - 2a. Not all syntactic boundaries appeared to be marked by prosodic boundaries.
   - 2b. The relationship between the location of a prosodic boundary found by the algorithm and the location of the corresponding syntactic constituent was far from precise. Depending on intonation contour, syllable stress, and phonetic effects, the prosodic boundary could occur either before or after the syntactic boundary.
3. Even assuming solutions to the above problems, any prosodic boundary location algorithm, including Lea's, will make a certain number of errors. The questions then are how much undertainty is tolerable for prosodics to be still useful and how can prosodics best be used in the process of speech understanding.

During the past year, we have been working with the Univac group on the latter two problems. This interaction led to some specific proposals on fitting a prosodics knowledge source using Lea's boundary detection algorithm into the HWIM system. Such a prosodics component was implemented and operationally checked out toward the end of the project, but there was no time to evaluate its contribution to the

speech understanding process. This section describes this recent work in prosodics.

The initial stage of the Univac-BBN effort was for the Univac group to study the operation of the current version of HWIM, particularly with respect to the grammar (SMALLGRAM at the time) and the operation of the Parser. The fact that at the time all aspects of the system (especially the grammar) were steadily being modified did not make their job any easier. However, the basic way the system worked remained coherent enough for their conclusions to remain valid. The Univac study resulted in one principal and several secondary proposals for using prosodic knowledge in HWIM [Lea, 1976]. Since there was little time remaining, we decided to adopt the principal proposal, as it appeared to offer practical solutions to the problems enumerated above.

The Univac group noted those arcs of SMALLGRAM that were expected to be accompanied by prosodic boundaries. Specifically, a prosodic boundary would be expected to occur immediately before any word that caused the arc to be taken, where "immediately before" remained to be defined precisely. This specification narrowed the uncertainty about where the prosodic boundaries were predicted to occur from the level of a constituent to the level of a word, and it related those predictions directly to our grammar.

On each prosodically marked arc of the grammar there is a call to the function PBDRY. PBDRY checks whether a prosodic boundary occurs to the left of the word that caused the arc to be taken. If a boundary was found where predicted, a positive score increment is added to a prosodic score that the Parser maintains for each parse path; if no boundary was found, a negative adjustment is made to the prosodic score. This prosodic score is then sent to the Control component, where it is given to each event associated with a word match that results from the accompanying syntactic prediction.

More often that not, a new word can be added to a theory in several different ways. That is, the existing set of parse paths can

be extended along several different arcs that can consume the new word. As many of these arcs may be marked prosodically, a theory may be assigned more than one prosodic score. The Parser only returns the best of these, since Control does not distinguish among parse paths.

The Univac grammar work resulted in about 1/6 of the SMALLGRAM arcs bearing prosodic boundary predictions. (Since the grammar was continually changing, we could not expect them to keep current on it. When we changed to BIGGRAM, we "transliterated" the prosodic predictions from SMALLGRAM, since there was not time for another interaction with the Univac group.) Even though only a small portion of the grammar is affected, it turns out that all but very short sentences generally contain at least one or two prosodic predictions.

The original Univac proposal was for the prosodic checks to be performed when events were put into the event queue [Lea, 1976]. Prosodic score adjustments could then have the effect of reordering the queue so as to decrease the possibility of processing incorrect events. While applying knowledge early is certainly desirable, examining the grammar for these prosodic checks is tantamount to parsing. Thus, it seemed more natural and economical to incorporate prosodic evaluation into the syntactic processing of the event, as described above.

We now describe the way in which the function PBDRY determines if a prosodic boundary is present "immediately before" a word. After RTIME (Vol. II, Sec. A) has computed the parameters from the incoming speech signal, the prosodics component is called. (This is presently a separate program, for experimental convenience, but conceptually it could be a subroutine within RTIME.) This routine calls BBN implementations of Univac's boundary detection and syllabification routines, BOUND3 and CHUNK, which use F_0 and energy in the band 60-3000 Hz to produce prosodic boundary and syllable locations. It then "expands" each prosodic boundary from specifying a single point to specifying an interval in time over which the boundary applies. Figure 1 illustrates the extent to which the ends of the "boundary

interval" are moved to the right and left so that the interval covers the two syllables nearest the boundary.

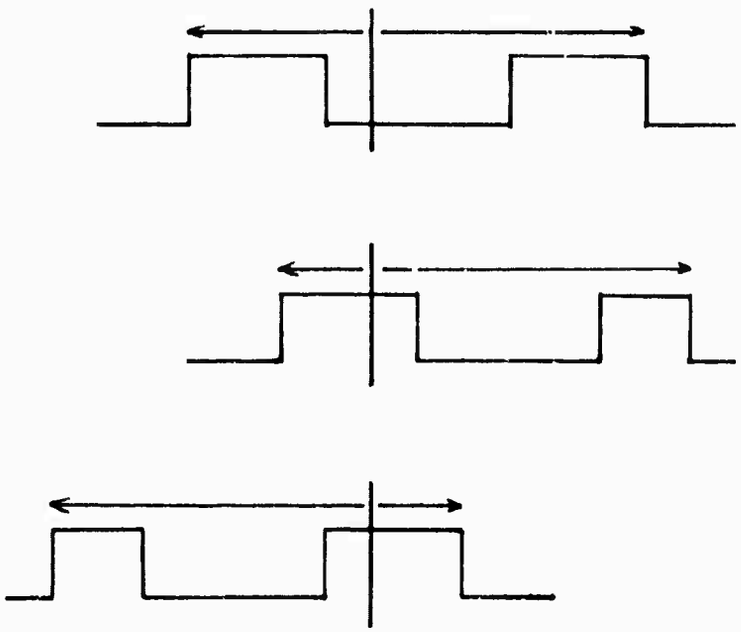


Fig. 1. Illustration of how the prosodic boundary location determined by BOUND3 (shown by the vertical line) is "expanded" to cover the nearest two syllables.

If the interval so defined includes a local maximum in F0 (also determined by BOUND3), the interval is truncated at that point, since the influence of a prosodic boundary caanot extend past it. Any boundaries found too near the beginning or end of the sentence to cover two syllables as shown in Fig. 1 are anomalous and therefore ignored. The times of the start and end of each boundary interval are then written on a .SBND file (for "spread boundaries") for subsequent use by the function PBDRY in the Parser.

In order to determine whether a word is "immediately preceded" by a prosodic boundary, PBDRY makes two checks:

a) If the starting time of the word falls within a boundary interval, then the answer is YES.

b) If the word does not carry primary stress on its first syllable (information available from the dictionary), then the prosodic boundary may occur within the word, in front of the first stressed syllable. In the current HWIM system, the times of the syllables are not available to the Parser, so a much less precise check is made: in the case of non-initial stress, if there is any overlap between the boundary interval and the word, the answer is YES.

c) Otherwise, the answer is NO.

The Univac group made estimates of +50 and -50 for the prosodic score adjustments, based on the sizes of scores they observed on HWIM trace listings. The likelihood ratio scoring method used in HWIM provides a rational basis for assigning such scores, based on prosodic component performance, as follows:

$$\frac{P(\text{boundary is found} \mid \text{arc predicts a boundary})}{P(\text{boundary is found})} \quad (1)$$

$$\frac{P(\text{no boundary is found} \mid \text{arc predicts a boundary})}{P(\text{no boundary is found})} \quad (2)$$

A first approximation may be made from performance measures cited by Lea, which do not take into account the particular boundary location to word location algorithms described above. Lea cites previous studies that show that about 90% of the expected boundaries are found; using this figure assumes that Univac's marking of our grammar works equally well. The denominator of (1) should be the proportion of boundaries found if they were sought on every word hypothesized, not just on those with marked grammar arcs. Lea's figure of 1/6 of the arcs of the grammar marked is certainly much too low an estimate of this. If we pick twice this value as a guess, then (1) and (2) become 0.9/0.33 = 2.7 and 0.1/0.67 = 0.15. To simplify arithmetic, we use 100 times the log of the likelihood ratio, so (1) and (2) translate to +44 and -82, respectively (not far from Univac's guesses). Obviously, performance estimates derived from our own implementation would produce better estimates, which would probably be smaller, due to the imprecision of the boundary spreading. (The BOUND3 routine also produces "confidence" estimates for its boundary locations, which could be factored into these probability estimates as well.)

Unfortunately, time did not permit us to carry the work on the prosodics component as far as testing its effectiveness in aiding the speech understanding process. In addition to that obvious task, there is more work on prosodics that should be done. Three topics specifically related to the work described above are:

1) Refinement of the grammar-based predictions of prosodic boundaries. Lea was unfamiliar with prosodic effects in some categories of our grammar (such as numbers and dates), so some of the predictions bear less confidence than others. Studies of travel budget sentences would lead to better predictions.

2) Refinement of the relationship of the locations of the prosodic boundaries and the grammatical boundaries. The "boundary expansion" and PBDRY checking algorithms described above are obviously rather imprecise.

3) More extensive testing of the prosodics component would lead to more accurate estimation of the proper values for scoring prosodic events.

Other possible extensions of prosodic knowledge to the speech understanding process may be found in [Lea, 1976].

References

[1] Bates, M. (1975a)
"The Use of Syntax in a Speech Understanding System," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-23, No. 1, February, pp. 112-117.

[2] Bates, M. (1975b)
"Syntactic Analysis in a Speech Understanding System," BBN Report No. 3116, Bolt Beranek and Newman Inc., Cambridge, Ma. 02138.

[3] Bates, M. and J. Wolf (1975)
"Prosodics," in Speech Understanding Systems, Quarterly Progress Report No. 2, pp. 25-37, BBN Report No. 3080, Bolt Beranek and Newman Inc., Cambridge, Ma. 02138.

[4] Burton, R.R. (1976)
"Semantic Grammar: A Technique for Efficient Language Understanding in Limited Domains," Ph.D. thesis, Dept. of Information and Computer Science, University of California at Irvine.

[5] Burton, R.R. and W.A. Woods (1976)
"A Compiling System for Augmented Transition Networks," in preprints of COLING-76, Ottawa, Canada, 28 June-2 July.

[6] Earley, J. (1970)
"An Efficient Context Free Parsing Algorithm," CACM, Vol. 13, No. 2, February.

[7] Lea, W.A., M.F. Medress and T.E. Skinner (1973)
"Prosodic Aids to Speech Recognition: II. Syntactic Segmentation and Stressed Syllable Location," Report No. PX10232, Sperry Univac, St. Paul, Minn.

[8] Lea, W.A. (1972)
"Intonational Cues to the Constituent Structure and Phonemics of Spoken English," Ph.D. thesis, Purdue University, Lafayette, Ind.

[9] Lea, W.A. (1973)
"An Approach to Syntactic Recognition Without Phonemics," IEEE Trans. Audio Electroacoustics AU-21, pp. 249-258.

[10] Lea, W.A. (1976)
"Prosodic Aids to Speech Recognition: VIII, Listeners' Perceptions of Selected English Stress Patterns," Report No. PX 11711, Sperry Univac, St. Paul, Minn.

[11] Woods, W.A., R.M. Kaplan, and B.L. Nash-Webber (1972)
"The Lunar Sciences Natural Language Information System: Final Report," BBN Report No. 2378, Bolt Beranek and Newman Inc., Cambridge, Ma. 02138 (available from NTIS as N72-23155).

[12] Woods, W.A. (1969)
"Augmented Transition Networks for Natural Language Analysis," Report No. CS-1, Aiken Computation Laboratory, Harvard University, December.

[13] Woods, W.A. (1970)
"Transition Network Grammars for Natural Language Analysis," CACM 13(10).

Appendix 1 - Listing of MIDGRAM

```
(AMOUNT/
   (JUMP AMOUNT/MOD )
   (CAT NUM-ADJ
        (SETP MOD (BUILDQ (ADJ *)))
        (TO AMOUNT/MOD))
   (WRD (LESS MORE)
        (SETP MOD (COND ((EQ LEX (QUOTE LESS))
                         (QUOTE (ADJ LESS#THAN)))
                        (T (QUOTE (ADJ GREATER#THAN)))))
        (TO AMOUNT/THAN?)))


(AMOUNT/AND?
   (WRD -PAUSE-
        (PBDRY)
        (TO AMOUNT/AND?))
   (WRD AND
        (PBDRY)
        (TO AMOUNT/CENTS?)))


(AMOUNT/CENTS!
   (WRD (CENT CENT-S)
        (SCOPE (AMOUNT/CENTS?)
               (VERIFY (OR (AND (IGREATERP (GETR CENTS)
                                           1)
                                (EQ (CATP LEX N NUMBER)
                                    (QUOTE PL)))
                           (AND (EQP (GETR CENTS)
                                     1)
                                (EQ (CATP LEX N NUMBER)
                                    (QUOTE SG)))))
        (TO AMOUNT/POP)))
```



```
(AMOUNT/CENTS?
   (PUSH NUM/
        (SETP CENTS (NUMBOP *))
        (SCOPE (AMOUNT/)
               (SETR NUM (PPLUS (GETR NUM)
                                (PQUOTIENT (GETP CENTS)
                                           100))))
        (TO AMOUNT/CENTS!)))


(AMOUNT/DOLLARS
   (JUMP AMOUNT/POP )
   (JUMP AMOUNT/AND? ))


(AMOUNT/MOD
   (PUSH NUMBER/
        (SCOPE (AMOUNT/)
               (COND ((NULLR MOD)
                      (PBDRY))))
        (VERIFY (NOP (GETP * DIGITS)
                     (EQ LEX (QUOTE OH))))
        (SETR NUM (NUMBOP *))
        (TO AMOUNT/NUM)))


(AMOUNT/NUM
   (WRD (DOLLAR DOLLAR-S BUCKS)
        (SCOPE (AMOUNT/)
               (VERIFY (OR (AND (IGREATERP (GETR NUM)
                                           1)
                                (OR (EQ (CATP LEX N NUMBER)
                                        (QUOTE PL))
                                    (EQ LEX (QUOTE BUCKS))))
                           (AND (EQP (GETR NUM)
                                     1)
                                (EQ (CATP LEX N NUMBER)
                                    (QUOTE SG))))))
        (SETR NU (CATP LEX N ONS))
        (TO AMOUNT/DOLLARS))
   (WRD K
        (SCOPE (AMOUNT/MOD)
               (SETR NUM (ITIMES (GETR NUM)
                                 1000)))
        (TO AMOUNT/POP)))
```

```
(AMOUNT/POP
   (POP (BUILDQ (@ ($)
                   (*)
                   @)
                NUM
                (COND ((GETR MOD)
                       (LIST (GETR MOD)))))
        (SETR ANT (GETR NUM))
        (LIFTR ANT)
        (LIFTR MOD)))


(AMOUNT/THAN?
   (WRD THAN
        (TO AMOUNT/MOD)))


(ARITH/
   (PUSH EXPENSE-DET/
        (SETR ARITH *)
        (TO ARITH/POP))
   (PUSH AMOUNT/
        (SETR ARITH *)
        (TO ARITH/POP))
   (PUSH NUMBER/
        (SETR ARITH *)
        (TO ARITH/POP)))


(ARITH/POP
   (POP (GETR ARITH) ))


(BI-PL/
   (WRD THE
        (SETR DET (BUILDQ (DET (ART *))))
        (TO BI/PLDET))
   (JUMP BI/PLDET ))


(BI-SG/
   (WRD (A AN THE)
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (ART *))))
        (SETR ARTAGREE T)
        (* ARTAGREE IS SET TO T TO INDICATE THAT ARTICLE AGREEMENT NEEDS
           TO BE DONE)
        (TO BI/SGDET))
   (WRD BUDGET
        (PBDRY)
        (TO BI/BUDGET))
   (JUMP BI/BUDGET ))
```



```
(BI/BI
   (JUMP BI/POP )
   (WRD UNDER
        (SETR PREP *)
        (TO BI/UNDER)))


(BI/BUDGET
   (WRD ITEM
        (SETRQ SEMHEAD BUDGET/ITEM)
        (TO BI/NUM?)))


(BI/ITEM?
   (WRD ITEM
        (SCOPE (BI/MOD)
               (AND (NULLR BUDGET)
                    (PBDRY)))
        (SCOPE (BI-SG/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (SETR HEAD (BUILDQ (N *)))
        (SETRQ NU SG)
        (TO BI/BI)))


(BI/ITEMS?
   (WRD ITEM-S
        (SETRQ SEMHEAD BUDGET/ITEM)
        (SETR HEAD (BUILDQ (N *)))
        (SETRQ NU PL)
        (TO BI/BI)))


(BI/MOD
   (WRD BUDGET
        (SETRQ BUDGET T)
        (SCOPE (BI/SGDET)
               (AND (NULLR PREMODS)
                    (PBDRY)))
        (SCOPE (BI-SG/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (TO BI/ITEM?))
   (JUMP BI/ITEM? ))
```

```
(BI/NUM?
   (PUSH NUM/
       (SETR ITEMNO (NUMBOF *))
       (TO BI/NUMPOP)))


(BI/NUMPOP
   (POP (BUILDQ (NP (ITEMNO *))
                ITEMNO)
       (SCOPE (BUDGET-ITEM/)
              (SETR INTERP (BUILDQ (! # # BUDGET/ITEM ((NUMBER *)))
                                   (OR (GETR SEMQUANT)
                                       (QUOTE THE))
                                   (GENSYM)
                                   ITEMNO))
              (LIFTR INTERP))))


(BI/PLDET
   (WRD BUDGET
       (PBDRY)
       (TO BI/ITEMS?))
   (WRD ITEM-S
       (PBDRY)
       (SETR HEAD (BUILDQ (N *)))
       (SETRQ NU PL)
       (TO BI/BI)))


(BI/POP
   (POP (NPBUILD)
       (SCOPE (BUDGET-ITEM/)
              (SETR INTERP (BUILDQ (! * # BUDGET/ITEM *)
                                   SEMQUANT
                                   (GENSYM
                                   SEMLINKS))
              (LIFTR INTERP))))


(BI/SGDET
   (WRD NEW
       (PBDRY)
       (ADDL SEMLINKS (QUOTE (MODALITY NEW,
       (COND ((GETR ARTAGREE)
              (VERIFY (ARTAGREE (GETR DET)
                                LEX))
              (SETR ARTAGREE NIL)))
       (ADDL PREMODS (BUILDQ (ADJ *)))
       (TO BI/MOD))
   (JUMP BI/MOD ))
```



```
(BI/NUMBER
   (PUSH BUDGET/
       (ADDL SEMLINKS (BUILDQ (ITEM/OF #)
                              (GETF INTERP)))
       (SCOPE (BI/BI)
              (ADDL PREMODS (BUILDQ (PP (PREP *)
                                        *)
                                    PREP)))
       (TO BI/POP)))


(BUDGET-ITEM-QUANT/
   (JUMP BUDGET-ITEM/ ))


(BUDGET-ITEM/
   (WRD ALL
       (SETR SEMQUANT *)
       (SETR DET (BUILDQ (DET (QUANT *))))
       (TO BI/PLDET))
   (JUMP BI-SG/ )
   (JUMP BI-PL/
       (SETRQ SEMQUANT ALL)))


(BUDGET/
   (WRD OVERHEAD
       (PBDRY)
       (SETRQ SEMQUANT THE)
       (SETRQ SEMHEAD DB/BUDGET)
       (ADDL SEMLINKS (QUOTE (BUDGET/OF OVERHEAD)))
       (SETRQ DET (DET))
       (SETR N *)
       (SETR HEAD (BUILDQ (N *)))
       (SETRQ NU SG)
       (TO BUDGET/POP))
   (WRD THE
       (SETR SEMQUANT *)
       (SETR DET (BUILDQ (DET (ART *))))
       (TO BUDGET/DET))
   (WRD THAT
       (VERIFY (ASKFORK (QUOTE TRIP)
                        (QUOTE (FOR: SOME X / (FINDQ: DB/BUDGET)
                                     : T : (IMPLIES? X)))))
       (SETR SEMQUANT *)
       (SETRQ DEFINITEDET T)
       (SETR DET (BUILDQ (DET *)))
       (TO BUDGET/DET))
```

```
;  <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:10


    (WRD (MY OUR)
        (ADDL SEMLINKS (BUILDQ (BUDGET/OF (! SOME # DB/CONTRACT
                                             ((GROUP #))))
                               (GENSYM)
                               (BUILDQ (! SOME # DB/GROUP
                                          ((MEMBERS #)))
                                       (GENSYM)
                                       (SEMPRO LEX))))
        (SETRQ NAMED T)
        (SETR DET (BUILDQ (DET (POSS *))))
        (TO BUDGET/DET)))


(BUDGET/BUDGET
   (JUMP BUDGET/POP
        (SCOPE (BUDGET/)
               (COND
                 ((NULLR NAMED)
                  (VERIFY (ASKFORK (QUOTE TRIP)
                                   (QUOTE (TEST: (FOR: SOME X /
                                                       (FINDQ: DB/BUDGET)
                                                       : T :
                                                       (INFOCUS? X)))))))
                 ))
        (WRD FOR
             (SCOPE (BUDGET/)
                    (VERIFY (NULLR PROJECTFLAG))
                    (VERIFY (NULLR DEFINITEDET)))
             (SETR PREP *)
             (TO BUDGET/FOR)))


(BUDGET/BUDGET?
   (WRD BUDGET
        (SCOPE (BUDGET/DET)
               (AND (NULLP PREMODS)
                    (PBDRY)))
        (SETRQ SEMREAD DB/BUDGET)
        (SCOPE (BUDGET/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (SETR HEAD (BUILDQ (N *)))
        (SETRQ NU SG)
        (TO BUDGET/BUDGET)))
```

```
;  <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:11


(BUDGET/DET
   (PUSH PROJECT/
        (ADDL SEMLINKS (BUILDQ (PROJECT #)
                               (GETP INTERP)))
        (SETRQ NAMED T)
        (SETR PROJECTFLAG T)
        (SCOPE (BUDGET/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (ADDL PREMODS *)
        (TO BUDGET/BUDGET?))
   (WRD (CURRENT NEW)
        (PBDRY)
        (ADDL SEMLINKS (BUILDQ (MODALITY *)))
        (SCOPE (BUDGET/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO BUDGET/BUDGET?))
   (JUMP BUDGET/BUDGET? ))


(BUDGET/FOR
   (CAT POSS
        (VERIFY (NOT (FMEMB LEX (QUOTE (WHOSE YOUR -S)))))
        (ADDL SEMLINKS
              (BUILDQ (BUDGET/OF (! SOME # DB/CONTRACT
                                    ((GROUP (! SOME # DB/GROUP
                                               ((MEMBERS #)))))))
                      (GENSYM)
                      (GENSYM)
                      (SEMPRO LEX)))
        (SETR PREMOD (BUILDQ (DET (POSS *))
                             LEX))
        (TO BUDGET/FOR-OUR))
   (PUSH THE-PROJECT/
        (ADDL SEMLINKS (BUILDQ (PROJECT #)
                               (GETP INTERP)))
        (SCOPE (BUDGET/BUDGET)
               (ADDL NMODS (BUILDQ (PP (PREP *)
                                       *)
                                   PREP)))
        (TO BUDGET/POP)))
```

```
(BUDGET/FOR-OUR
   (WRD PROJECT-N
        (PBDRY)
        (SCOPE (BUDGET/BUDGET)
               (ADDL NMODS (BUILDQ (PP (PREP *)
                                       (NP * (N *)))
                                   PREP PREMOD)))
        (TO BUDGET/POP)))


(BUDGET/NUMBER
   (PUSH NUM/
        (SETR BUDGETNUM (NUMBOF *))
        (TO BUDGET/NUMBERPOP)))


(BUDGET/NUMBER?
   (WRD NUMBER
        (TO BUDGET/NUMBER)))


(BUDGET/NUMBERPOP
   (POP (BUILDQ (NP (BUDGETNUM *))
                BUDGETNUM) ))


(BUDGET/POP
   (POP (NPBUILD)
        (SCOPE (BUDGET/)
               (SETR INTERP (BUILDQ (! * # * *)
                                    SEMQUANT
                                    (GENSYM)
                                    SEMHEAD SEMLINKS))
               (LIFTR INTERP))))


(CITY/
   (CAT CITY
        (PBDRY)
        (ADDL SEMLINKS (BUILDQ (CITY *)))
        (SETR CITY *)
        (SETRQ LOCATION T)
        (TO CITY/CITY))
   (CAT STATE
        (PBDRY)
        (* IF PUSHED TO FROM MEETING/MOD, PRECEDING NEXT, FIRST, ETC
           WILL GET BDRY INSTEAD OF THIS. THIS FACT IS BEING IGNORED.)
        (SETRQ LOCATION T)
        (COND ((CATCHECK (WORD *)
                         (QUOTE CITY))
               (* AMBIGUOUS LOCATION)
               (SETR AMBIG *)
               (ADDL SEMLINKS (BUILDQ (AMBIG *))))
              (T (SETR STATE *)
                 (ADDL SEMLINKS (BUILDQ (STATE *)))))
        (TO CITY/POP)))
```



```
   (CAT COUNTRY
        (PBDRY)
        (ADDL SEMLINKS (BUILDQ (COUNTRY *)))
        (SETRQ LOCATION T)
        (SETR COUNTRY *)
        (TO CITY/POP))
   (WRD THE
        (SETRQ LOCATION T)
        (TO CITY/THE)))


(CITY/CITY
   (JUMP CITY/POP
        (* IF CITY IS ALSO A STATE, PREVIOUS STATE ARC WILL GET IT AS
           AMBIGUOUS)
        (SCOPE (CITY/)
               (VERIFY (NOT (CATCHECK (WORD (GETR CITY))
                                      (QUOTE STATE))))))
   (CAT STATE
        (ADDL SEMLINKS (BUILDQ (STATE *)))
        (SCOPE (CITY/)
               (VERIFY (GOODPLACE (GETR CITY)
                                  LEX)))
        (SETR STATE *)
        (TO CITY/POP))
   (CAT COUNTRY
        (ADDL SEMLINKS (BUILDQ (COUNTRY *)))
        (SCOPE (CITY/)
               (VERIFY (GOODPLACE (GETR CITY)
                                  LEX)))
        (SETR COUNTRY *)
        (TO CITY/POP)))


(CITY/POP
   (POP (NPRBUILD)
        (SCOPE (CITY/)
               (SETR INTERP (BUILDQ (! THE * LOCATION *)
                                    (GENSYM)
                                    SEMLINKS))
               (LIFTR INTERP)
               (AND (GETR THEFEAT)
                    (LIFTR THEFEAT)))))


(CITY/THE
   (WRD (EAST WEST)
        (PBDRY)
        (ADDL SEMLINKS (BUILDQ (COAST *)))
        (SETR ADJ LEX)
        (TO CITY/THE-COAST?))
   (JUMP CITY/THE-COAST?
        (ADDL SEMLINKS (BUILDQ (COAST AMBIG)))))
```

```
(CITY/THE-COAST
   (WRD COAST
        (SCOPE (CITY/THE)
               (AND (NULLR ADJ)
                    (PBDRY)))
        (SETR THEFEAT T)
        (SETR AMBIG (COND ((GETR ADJ)
                           (BUILDQ (* #)
                                   ADJ LEX))
                          (T LEX)))
        (TO CITY/POP)))


(CONTRACT/
   (WRD THE
        (SETR DET (BUILDQ (ART *)))
        (TO CONTRACT/DET))
   (WRD (ODR MY HIS HER)
        (ADDL SEMLINKS (BUILDQ (GROUP (! SOME # IN/GROUP
                                         ((MEMBERS #))))
                               (GENSYM)
                               (SEMPRO LEX)))
        (SETR DET (BUILDQ (DET (POSS *))))
        (TO CONTRACT/DET)))


(CONTRACT/DET
   (JUMP CONTRACT/SPONSOR )
   (PUSH PROJECT/
        (ADDL SEMLINKS (BUILDQ (PROJECT #)
                               (GETF INTERP)))
        (ADDL PREMODS *)
        (TO CONTRACT/SPONSOR)))


(CONTRACT/GROUP
   (WRD CONTRACT
        (PBDRY)
        (SETR HEAD (BUILDQ (N *)))
        (SETRQ ND SG)
        (TO CONTRACT/POP)))


(CONTRACT/POP
   (POP (NPBUILD)
        (SCOPE (CONTRACT/)
               (SETR INTERP (BUILDQ (! THE # DB/CONTRACT *)
                                    (GENSYM)
                                    SEMLINKS))
               (LIFTR INTERP))))
```



```
(CONTRACT/SPONSOR
   (JUMP CONTRACT/GROUP )
   (WRD GRODP
        (TO CONTRACT/GRODP)))


(DATEMOD-EXTENT/
   (JUMP DATEMOD/ADJ?
        (SETRQ PREP DURING))
   (WRD (SINCE BEFORE AFTER DURING TILL UNTIL)
        (SETR PREP LEX)
        (TO DATEMOD/ADJ?))
   (WRD (IN SINCE BEFORE AFTER DURING)
        (SETR PREP LEX)
        (TO DATEMOD/IN))
   (WRD FOR
        (SETR PREP LEX)
        (TO DATEMOD/FOR)))


(DATEMOD/
   (WRD (TODAY YESTERDAY TOMORROW)
        (SETRQ PREP ON)
        (SETR DAY *)
        (TO DATEMOD/POP))
   (WRD ON
        (SETR PREP *)
        (TO DATEMOD/THIS))
   (WRD ON
        (SETR PREP *)
        (TO DATEMOD/ON))
   (WRD (THIS LAST NEXT)
        (SETRQ PREP ON)
        (SETR DADJ *)
        (TO DATEMOD/THIS))
   (JUMP DATEMOD-EXTENT/ ))


(DATEMOD/ADJ
   (CAT MONTH
        (SCOPE (DATEMOD/IN)
               (AND (NULLR DADJ)
                    (PBDRY)))
        (SETR MONTH *)
        (TO DATEMOD/IN-MONTH))
   (PUSH YEAR-DATE/
        (SETR YEAR *)
        (TO DATEMOD/POP)))
```

```
(DATEMOD/ADJ?
   (WRD (THIS LAST NEXT)
        (AND (NOT (EQ LEX (QUOTE THIS)))
             (PBDRY))
        (SCOPE (DATEMOD-EXTENT/)
               (VERIFY (NOT (AND (EQ (WORD (GETR PREP))
                                     (QDOTE SINCE))
                                 (EQ LEX (QUOTE NEXT))))))
        (SETR DADJ *)
        (TO DATEMOD/LAST)))


(DATEMOD/DAY
   (JUMP DATEMOD/ON ))


(DATEMOD/FOR
   (PUSH NUM/
        (SETR DADJ (NUMBOF *))
        (TO DATEMOD/FOR-NUM))
   (WRD A
        (SETRQ DADJ 1)
        (TO DATEMOD/FOR-NUM))
   (WRD (THIS LAST NEXT)
        (AND (NOT (EQ LEX (QDOTE THIS)))
             (PBDRY))
        (SETR DADJ *)
        (TO DATEMOD/FOR-NUM)))


(DATEMOD/FOR-NUM
   (CAT DURATION
        (SCOPE (DATEMOD/FOR)
               (AND (OR (EQ (GETR DADJ)
                            1)
                        (EQ (GETR DADJ)
                            THIS))
                    (PBDRY)))
        (SCOPE (DATEMOD/FOR)
               (VERIFY (COND ((AND (NUMBERP (GETR DADJ))
                                   (IGREATERP (GETR DADJ)
                                              1))
                              (EQ (QUOTE PL)
                                  (CATF LEX DURATION NUMBER)))
                             (T T)))
               (VERIFY (NOT (AND (OR (EQ (GETR DADJ)
                                         1)
                                     (FMEMB (CAR (GETR DADJ))
                                            (QDOTE (THIS LAST NEXT))))
                                 LEX
                                 (NOT (FMEMB LEX
                                             (QUOTE (WEEK DAY MONTH
                                                          QUARTER YEAR)))
                                      ))))
```



```
                (SETR TIME LEX)
                (SETR DADJ (BUILDQ (ADJ *)
                                   DADJ)))
        (TO DATEMOD/POP)))


(DATEMOD/IN
   (WRD THE
        (SETR DADJ (BUILDQ (DET (ART THE))))
        (TO DATEMOD/IN-THE))
   (WRD (LATE EARLY)
        (PBDRY)
        (SETR DADJ *)
        (TO DATEMOD/ADJ))
   (JUMP DATEMOD/ADJ ))


(DATEMOD/IN-MONTH
   (JUMP DATEMOD/POP )
   (WRD OF
        (TO DATEMOD/YEAR?)))


(DATEMOD/IN-THE
   (CAT SEASON
        (PBDRY)
        (SETR TIME *)
        (TO DATEMOD/POP))
   (WRD (NEXT LAST)
        (PBDRY)
        (SETR DADJ *)
        (TO DATEMOD/IN-THE-LAST)))


(DATEMOD/IN-THE-LAST
   (CAT DDRATION
        (SETR TIME *)
        (TO DATEMOD/POP)))


(DATEMOD/LAST
   (CAT MONTH
        (SETR MONTH *)
        (TO DATEMOD/POP))
   (CAT SEASON
        (SETR TIME *)
        (TO DATEMOD/POP))
   (CAT DURATION
        (SCOPE (DATEMOD/ADJ?)
               (AND (EQ DADJ (QUOTE THIS))
                    (PBDRY)))
        (VERIFY (AND (EQ (QUOTE SG)
                         (CATF LEX DURATION NUMBER))
                     (NEQ LEX (QDOTE DAY))))
        (SETR TIME *)
        (TO DATEMOD/POP)))
```

```
(DATEMOD/MONTH
   (PUSH YEAR-DATE/
        (SCOPE (DATEMOD/ DATMOD-POINT/ DATEMOD-EXTENT/)
               (VERIFY (NULLR DADJ)))
        (SETR YEAR *)
        (TO DATEMOD/POP))
   (JUMP DATEMOD/POP ))


(DATEMOD/MONTH?
   (CAT MONTH
        (PBDRY)
        (SETR MONTH *)
        (TO DATEMOD/MONTH)))


(DATEMOD/ON
   (CAT MONTH
        (PBDRY)
        (SCOPE (DATEMOD/)
               (VERIFY (NULLR DADJ)))
        (SETR MONTH *)
        (TO DATEMOD/ON-MONTH))
   (WRD THE
        (TO DATEMOD/ON-THE)))


(DATEMOD/ON-MONTH
   (PUSH ORD/
        (VERIFY (ILESSP (NUMBOF *)
                        32))
        (SETR DAY (NUMBOF *))
        (TO DATEMOD/MONTH))
   (PUSH NUM/
        (VERIFY (ILESSP (NUMBOF *)
                        32))
        (SETR DAY (NUMBOF *))
        (TO DATEMOD/MONTH)))


(DATEMOD/ON-THE
   (PUSH ORD/
        (SETR DAY (NUMBOF *))
        (TO DATEMOD/ON-THE-DAY)))


(DATEMOD/ON-THE-DAY
   (WRD OF
        (TO DATEMOD/MONTH?))
   (JUMP DATEMOD/POP ))
```



```
(DATEMOD/POP
   (POP (BUILDQ (PP (PREP *)
                    *)
                PREP
                (DATEBUILD))
        (SCOPE T (SETR INTERP (DATEBUILD)))
        (LIFTR INTERP)))


(DATEMOD/THIS
   (CAT WEEKDAY
        (PBDRY)
        (SETR WEEKDAY *)
        (TO DATEMOD/POP))
   (CAT WEEKDAY
        (PBDRY)
        (SETR WEEKDAY *)
        (TO DATEMOD/ON))
   (CAT MONTH
        (PBDRY)
        (SCOPE (DATEMOD/)
               (VERIFY (GETR NOPREP)))
        (SETR MONTH *)
        (TO DATEMOD/POP)))


(DATEMOD/YEAR?
   (PUSH YEAR-DATE/
        (SETR YEAR *)
        (TO DATEMOD/POP)))


(DIGIT-STRING/
   (CAT DIGITS
        (PBDRY)
        (SETR NUMSTR (NUMBOF *))
        (TO DIGITS/D)))
```

```
(DIGITS/D
   (POP (GETR NUMSTR)
        (VERIFY (COND ((CATCHECK LEX (QUOTE INTEGER))
                       (INRANGE (NUMBOF LEX 0 9)))
                      (T T)))
        (SCOPE (DIGIT-STRING/)
               (* ACCOUNT NUMBERS SHOULD ALSO BE ALLOWED)
               (SELECTQ (LENGTH (UNPACK (GETR NUMSTR)))
                        (4 (SETRQ DIGITSTRINGFEATS TRIPNO)
                           (LIFTR DIGITSTRINGFEATS))
                        ((5 6)
                         (SETRQ DIGITSTRINGFEATS JCBNO)
                         (LIFTR DIGITSTRINGFEATS))
                        (VERIFY NIL))))
   (CAT DIGITS
        (VERIFY (NOT (IGREATERP (LENGTH (UNPACK (GETR NUMSTR)))
                                5)))
        (SCOPE (DIGIT-STRING/)
               (SETR NUMSTR (IPLUS (NUMBOF *)
                                   (ITIMES 10 (GETR NUMSTR)))))
        (TO DIGITS/D))
   (WRD OH
        (VERIFY (NOT (IGREATERP (LENGTH (UNPACK (GETR NUMSTR)))
                                5)))
        (SCOPE (DIGIT-STRING/)
               (SETR NUMSTR (ITIMES 10 (GETR NUMSTR))))
        (TO DIGITS/D)))


(ERD/
   (POP (GETR INTERP)
        (VERIFY (RIGHTENDP))))


(EX-ALL-OF?/
   (WRD OF
        (TO EX-PL-NON-DEM/)))


(EX-DET-PL-DEF/
   (WRD ALL
        (PBDRY)
        (SETRQ SEMQUANT ALL)
        (SETR DET (BUILDQ (DET (QUANT *))))
        (TO EX-PL/DET))
   (WRD ALL
        (PBDRY)
        (SETR SUPERDET (BUILDQ (DET (QUANT *))))
        (TO EX-PL-NOR-DEM/))
   (WRD (ALL SOME ANY)
        (COND ((EQ LEX (QUOTE ALL))
               (SETRQ SEMQUANT ALL))
              (T (SETRQ SEMQUANT ANY)))
        (SETR SUPERDET (BUILDQ (DET (QUANT *))))
        (TO EX-ALL-OF?/))
   (JUMP EX-PL-NOR-DEM/ ))
```



```
(EX-DET-PL-INDEF/
   (WRD (SOME ANY)
        (SETRQ SEMQUANT ANY)
        (SETR DET (BUILDQ (DET (QUANT *))))
        (TO EX-PL/DET))
   (JUMP EX-PL/DET
        (SETRQ SEMQUANT ANY)))


(EX-DET-PL/
   (JUMP EX-DET-PL-DEF/ )
   (JUMP EX-DET-PL-INDEF/ ))


(EX-DET-SG-DEF/
   (WRD (THIS THAT)
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (FOR: SOME X / (FIRDQ: DB/EXPENSE)
                                      : T : (IRPCCUS? X)))))
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET *)
                          LEX))
        (TO EX-SG/DET))
   (WRD THE
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (ART *))
                          LEX))
        (TO EX-SG/DET))
   (CAT POSS
        (SETRQ SEMQUART THE)
        (INCORP SEMLINKS (BUILDQ (TRAVELER *)
                                 (SEMPRO LEX)))
        (SETR DET (BUILDQ (DET (POSS *))
                          LEX))
        (TO EX-SG/DET))
   (PUSH NAME/
        (SETRQ SEMQUART THE)
        (INCORP SEMLINKS (BUILDQ (TRAVELER *)
                                 (GETP INTERP)))
        (SETR PERSON *)
        (TO EX-SG-POSS?/)))
```

```
; <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:22

(EX-DET-SG-INDEF/
   (WRD (A AN)
        (SETRQ SEMQUANT SOME/1)
        (* THIS MEANS THE QUANT SHOULD BE SOME IN A QUESTION, ONE IN A
           DECLARATIVE)
        (SETR ARTAGREE T)
        (SETR DET (BUILDQ (DET (ART #))
                          LEX))
        (TO EX-SG/DET))
   (WRD ANY
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (QUANT #))
                          (NORD *)))
        (TO EX-SG/DET)))


(EX-DET-SG/
   (JUMP EX-DET-SG-DEF/ )
   (JUMP EX-DET-SG-INDEF/ ))


(EX-PL-NOM-DEM/
   (WRD (THESE THOSE)
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET #)
                          LEX))
        (TO EX-PL/DET))
   (WRD THE
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (ART #))
                          LEX))
        (TO EX-PL/DET))
   (CAT POSS
        (SETRQ SEMQUANT ALL)
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (SEMPRO LEX)))
        (SETR DET (BUILDQ (DET (POSS #))
                          LEX))
        (TO EX-PL/DET))
   (PUSH NAME/
        (SETRQ SEMQUANT ALL)
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (GETP INTERP)))
        (SETR PERSON *)
        (TO EX-PL-POSS?/)))
```

```
; <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:23

(EX-PL-POSS?/
   (WRD -S
        (SCOPE (EX-DET-PL-DEF/)
               (COND ((NULL DET)
                      (SETRQ DET (DET THE)))))
        (SCOPE (EX-DET-PL-DEF/)
               (ADDL WMODS (BUILDQ (PP (PREP BY)
                                       (NP *))
                                   PERSON)))
        (TO EX-PL/DET)))


(EX-PL/DET
   (JUMP EXPENSE/DET
        (SETRQ DETNUM PL)))


(EX-SG-POSS?/
   (WRD -S
        (SCOPE (EX-DET-SG-DEF/)
               (ADDL WMODS (BUILDQ (PP (PREP BY)
                                       (NP *))
                                   PERSON)))
        (TO EX-SG/DET)))


(EX-SG/DET
   (JUMP EXPENSE/DET
        (SETRQ DETNUM SG)))


(EXPENSE-DET/
   (JUMP EX-DET-PL/ )
   (JUMP EX-DET-SG/ ))


(EXPENSE/ADJ1
   (WRD (ESTIMATE-V-ED ACTUAL)
        (SCOPE (EX-DET-SG-INDEF/ EX-DET-SG-DEF/ EX-SG-POSS?/ EX-PL/DET)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (ADDL PREMODS (BUILDQ (ADJ (: FEATURES)
                                   *)))
        (TO EXPENSE/ADJ2))
   (JUMP EXPENSE/ADJ2 ))
```

```
(EXPENSE/ADJ2
   (WRD (COST COST-S EXPENSE EXPENSE-S PRICE)
        (SETRQ SEMHEAD DB/COST)
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (SETR HEAD (BUILDQ (N *)
                           (ROOTF * N)))
        (SETR NU (CATF LEX N NUMBER))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))
        (SCOPE (EX-DET-PL-DEF/ EX-PL-NON-DEF/ EX-DET-PL-INDEF/
                               EX-DET-SG-INDEF/ EX-DET-SG-DEF/)
               (VERIFY (GETR PREMODS))
               (SETR PREMODS (REVERSE (GETR PREMODS))))
        (TO *EXPENSE/COST))
   (WRD BUDGET
        (SETRQ SEMHEAD DB/AMOUNT)
        (ADDL SEMLINKS (BUILDQ (IN/BUDGET (! THE * DB/BUDGET))
                               (GENSYM)))
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (SETR HEAD (BUILDQ (N *)
                           (ROOTF * N)))
        (SETR NU (CATF LEX N NUMBER))
        (TO EXPENSE/BUDGET)))


(EXPENSE/BUDGET
   (WRD (FIGURE FIGURE-S)
        (SCOPE (EXPENSE/DET)
               (ADDL PREMODS (BUILDQ (ADJ *)
                                     (CADR (GETR HEAD))))
               (SETR PREMODS (REVERSE (GETR PREMODS))))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (SETR HEAD (BUILDQ (N *)
                                  (ROOTF * N)))
               (SETR NU (CATF LEX N NUMBER))
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))
        (TO EXPENSE/POP))
   (JUMP EXPENSE/POP
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))
        (SCOPE (EX-DET-PL-DEF/ EX-PL-NON-DEF/ EX-DET-PL-INDEF/
                               EX-DET-SG-INDEF/ EX-DET-SG-DEF/)
               (VERIFY (GETR PREMODS))
               (SETR PREMODS (REVERSE (GETR PREMODS))))))
```



```
(EXPENSE/COST
   (WRD (POP OF)
        (SETR PREP *)
        (TO EXPENSE/COST-OF))
   (PUSH DATAMOD/
        (INCORP SEMLINKS (GETP INTERP))
        (ADDL MMODS *)
        (TO EXPENSE/POP))
   (JUMP EXPENSE/POP ))


(EXPENSE/COST-OF
   (PUSH T-DET-DEF-OR-SG/
        (ADDL SEMLINKS (BUILDQ (COST/OF *)
                               (GETP INTERP)))
        (SCOPE (EX-DET-SG/ EX-DET-PL/ EX-PL-NON-DEF/ EX-DET-SG-DEF/)
               (ADDL MMODS (BUILDQ (PP (PREP *)
                                       *)
                                   PREP)))
        (TO EXPENSE/POP)))


(EXPENSE/DET
   (WRD REGISTRATION
        (PBDRY)
        (SETRQ SEMHEAD DB/FEE)
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (VERIFY (NULLP ARTAGREE)))
        (ADDL SEMLINKS (QUOTE (TYPE REGISTRATION)))
        (SETR HEAD (BUILDQ (N *)
                           (ROOTF * N)))
        (SETR NU (CATF LEX N NUMBER))
        (TO EXPENSE/REGISTRATION))
   (WRD (ROUND-TRIP ONE-WAY)
        (PBDRY)
        (ADDL SEMLINKS (BUILDQ (TYPE *)))
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO EXPENSE/MOD))
   (JUMP EXPENSE/MOD
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (VERIFY (NULLR ARTAGREE)))))
```

```
   (WRD PERDIEM
        (PBDRY)
        (SETRQ SEMHEAD PERDIEM)
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (SETR HEAD (BUILDQ (W *)))
        (SETR NU (CATP LEX W NUMBER))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETP DETNDM))))
        (TO EXPENSE/PER-DIEM))
   (WRD (TOTAL WHOLE ENTIRE TYPICAL)
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (VERIFY (NULLR ARTAGREE)))
        (ADDL PRENODS (BUILDQ (ADJ (: FEATDNES)
                                   *)))
        (TO EXPENSE/ADJ1))
   (WRD (COST COST-S EXPENSE EXPENSE-S PRICE)
        (PBDRY)
        (SETRQ SEMHEAD DB/COST)
        (SCOPE (EX-DET-SG-INDEF/ EX-PL/DET EX-DET-SG-DEF/ EX-SG-POSS?/)
               (VERIFY (NULLR ARTAGREE)))
        (SETR HEAD (BUILDQ (W #)
                           (ROOTP * W)))
        (SETR NU (CATP LEX W NUMBER))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNDM))))
        (TO EXPENSE/COST))
   (JUMP EXPENSE/ADJ1 ))


(EXPENSE/FARE
   (WRD FROM
        (TO EXPENSE/FARE-FROM))
   (JUMP EXPENSE/FARE-FROM-CITY
        (ADDL SEMLINKS (BUILDQ (STARTING/POINT BOSTON))))
   (JUMP EXPENSE/POP )
   (WRD THERE
        (ADDL SEMLINKS (BUILDQ (DESTINATION *)))
        (ADDL NMODS (BUILDQ (PP (PREP TO)
                                (NP (PRO THERE)
                                    (FEATS (ND SG);;;;;
        (TO EXPENSE/POP)))
```



```
(EXPENSE/FARE-FROM
   (PUSH CITY/
        (SETR FROM *)
        (ADDL SEMLINKS (BUILDQ (STARTING/POINT #)
                               (GETP INTERP)))
        (ADDL NMODS (BUILDQ (PP (PREP FROM)
                                (NP *))))
        (TO EXPENSE/FARE-FROM-CITY)))


(EXPENSE/FARE-FROM-CITY
   (WRD TO
        (TO EXPENSE/FARE-TO)))


(EXPENSE/FARE-TO
   (PUSH CITY/
        (SCOPE (EXPENSE/FARE-FROM EXPENSE/FARE)
               (VERIFY (NOT (EQUAL (GETR FROM)
                                   *))))
        (ADDL SEMLINKS (BUILDQ (DESTINATION #)
                               (GETP INTERP)))
        (ADDL NMODS (BUILDQ (PP (PREP TO)
                                (NP *))))
        (TO EXPENSE/POP)))


(EXPENSE/FEE
   (JUMP EXPENSE/POP
        (* CHECK THAT A CONFERENCE HAS BEEN MENTIONED)
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (TEST: (FOR: SOME X /
                                             (FINDQ: DB/CONFERENCE)
                                             : T ; (INFOCUS? X))))))
        (SCOPE T (SETR INTERP (BUILDQ (! * # * *)
                                      SEMQUANT
                                      (GENSYM)
                                      SEMHEAD SEMLINKS))))
   (WRD FOR
        (TO EXPENSE/FEE-FOR)))


(EXPENSE/FEE-FOR
   (PUSH MEETING/
        (ADDL SEMLINKS (BUILDQ (FEE/OF #)
                               'GETP INTERP)))
        (ADDL NMODS (BUILDQ (PP (PREP FOR)
                                *)))
        (TO EXPENSE/POP)))
```

```
(EXPENSE/MOD
   (JUMP EXPENSE/MOD-ADJ )
   (RMD (BUS TRAIN PLANE AIR)
        (ADDL SEMLINKS (BUILDQ (MODE/OF/TRANSPORT *)))
        (SETRQ SEMHEAD DB/FARE)
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (SCOPE (EX-DET-SG-INDEF/ EX-DET-SG-DEF/ EX-SG-POSS?/ EX-PL/DET)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (TO EXPENSE/MOD-ADJ)))


(EXPENSE/MOD-ADJ
   (RMD (FARE FARE-S)
        (SETRQ SEMHEAD DB/FARE)
        (SETR HEAD (BUILDQ (N *)
                           (ROOTP * FARE)))
        (SETR NU (CATP LEX FARE NUMBER))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))
        (SCOPE (EX-DET-SG-INDEF/ EX-DET-SG-DEF/ EX-SG-POSS?/ EX-PL/DET)
               (COND ((GETR ARTAGREE)
                      (VERIFY (ARTAGREE (GETR DET)
                                        LEX))
                      (SETR ARTAGREE NIL))))
        (TO EXPENSE/FARE)))


(EXPENSE/PER-DIEM
   (RMD IN
        (TO EXPENSE/PER-DIEM-IN)))


(EXPENSE/PER-DIEM-IN
   (PUSH CITY/
        (ADDL SEMLINKS (BUILDQ (LOCATION *)
                               (GETP INTERP)))
        (ADDL MMODS (BUILDQ (PP (PREP IN)
                                (NP *))))
        (TO EXPENSE/POP)))
```



```
(EXPENSE/POP
   (POP (COND ((NULLR SUPERDET)
               (NPBUILD))
              (T (BUILDQ (NP * (N (PPO ONE)
                                  (PP (PREP OF)
                                      *)
                                  (FEATS (NO PL)))
                             SUPERDET
                             (NPBUILD))))
        (SCOPE (EXPENSE/DET)
               (SETR SEMLINKS (CLEANSEM (GETR SEMLINKS)))
               (SETR INTERP (BUILDQ (! * * *)
                                    SEMQUANT
                                    (GENSYM)
                                    SEMHEAD SEMLINKS))
               (LIFTR INTERP))))


(EXPENSE/REGISTRATION
   (RMD FEE
        (SCOPE (EXPENSE/DET)
               (SETR HEAD (BUILDQ (N *)
                                  (ROOTP * N)))
               (SETR NU (CATP LEX N NUMBER)))
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))
        (ADDL PREMODS (BUILDQ (ADJ (* REGISTRATION))))
        (TO EXPENSE/FEE))
   (JUMP EXPENSE/FEE
        (SCOPE (EX-PL/DET EX-SG/DET)
               (VERIFY (EQ (GETR NU)
                           (GETR DETNUM))))))


(GROUP/
   (RMD THE
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (ART *))))
        (TO GROUP/DET)))


(GROUP/DET
   (JUMP GROUP/GROUP? )
   (RMD SUB
        (PBDET)
        (ADDL SEMLINKS (QUOTE (GROUP SUB)))
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO GROUP/GROUP?))
   (PUSH PROJECT/
        (ADDL SEMLINKS (BUILDQ (GROUP *)
                               (GETP INTERP)))
        (ADDL PREMODS *)
        (TO GROUP/GROUP?)))
```

```
;  <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:30


(GROUP/GROUP?
   (WRD (PROJECT-M GROUP)
        (SETRQ SEMHEAD DB/GROUP)
        (SETR HEAD (BUILDQ (M *)))
        (SETRQ NU SG)
        (TO GROUP/POP)))


(GROUP/POP
   (POP (MPBUILD)
        (SCOPE (GROUP/)
               (SETR INTERP (BUILDQ (! + @ + +)
                                    SEMQUANT
                                    (GENSYM)
                                    SEMHEAD SEMLINKS))
               (LIFTR INTERP))))


(MEETING/
   (WRD THE
        (SETR DET (BUILDQ (DET (ART @))
                          LEX))
        (TO MEETING/THE))
   (WRD THAT
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (FOR: SOME X / (FINDQ: DB/CONFERENCE)
                                      : T : (INFCCUS? X)))))
        (SETN SEMQUANT *)
        (SETR DET (BUILDQ (DET *)))
        (TO MEETING/SPONSOR))
   (CAT SPONSOR
        (PBODY)
        (* EVENTUALLY THIS ARC AND THE CAT MPR ABC SHOULD BE REDUCED TO
           A CAT SITE ABC)
        (SETRQ SEMQUANT THE)
        (ADDL SEMLINKS (BUILDQ (SPONSOR *)))
        (SETRQ SEMHEAD DB/CONFERENCE)
        (SETRQ NU SG)
        (SETR HEAD (BUILDQ (M *)))
        (TO MEETING/MEETING))
   (CAT MPR
        (PBODY)
        (* WILL PICK UP "CARNEGIE" , "UNIVAC" ,ETC.)
        (SETRQ MPR T)
        (SETRQ SEMQUANT THE)
        (ADDL SEMLINKS (BUILDQ (LOCATION *)))
        (SETRQ SEMHEAD DB/CONFERENCE)
        (SETRQ NU SG)
        (SETR HEAD (BUILDQ (M *)))
        (TO MEETING/MEETING)))
```

```
;  <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:31


(MEETING/IN
   (PUSH CITY/
        (ADDL SEMLINKS (BUILDQ (LOCATION @)
                               (GETF INTERP)))
        (ADDL MMODS (BUILDQ (PP (PREP IN)
                                *)))
        (TO MEETING/MEETING)))


(MEETING/MEETING
   (POP (MPBUILD)
        (SCOPE (MEETING/)
               (SETR INTERP (BUILDQ (! + @ + +)
                                    SEMQUANT
                                    (GENSYM)
                                    SEMHEAD SEMLINKS))
               (LIFTR INTERP)))
   (WRD IN
        (SCOPE (MEETING/MOD MEETING/)
               (VERIFY (NOT (GETF LOCATION)))
               (SETRQ LOCATION T))
        (TO MEETING/IN))
   (PUSH DATEMOD/
        (VERIFY (NOT (FIND (QUOTE SINCE)
                           (UNHASH *))))
        (SCOPE (MEETING/)
               (VERIFY (NULLR MPR)))
        (SCOPE (MEETING/THE)
               (VERIFY (NULLR TIME))
               (SETRQ TIME T)
               (ADDL SEMLINKS (GETF INTERP))
               (ADDL MMODS *))
        (TO MEETING/MEETING)))


(MEETING/MOD
   (CAT SPONSOR
        (SCOPE (MEETING/THE)
               (ANU (NULLR PREMODS)
                    (PBODY)))
        (ADDL SEMLINKS (BUILDQ (SPONSOR *)))
        (ADDL PREMODS (BUILDQ (ADJ (MP (MPR *))))))
        (TO MEETING/SPONSOR))
   (PUSH CITY/
        (VERIFY (NULL (GETF THEFEAT)))
        (ADDL SEMLINKS (BUILDQ (LOCATION @)
                               (GETF INTERP)))
        (ADDL PREMODS (BUILDQ (ADJ (MP *))))
        (SETRQ LOCATION T)
        (TO MEETING/SPONSOR))
   (JUMP MEETING/SPONSOR
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (FOR: SOME X / (FINDQ: DB/SPONSOR)
                                      : T ; (INFCCUS? X)))))))
```

```
(MEETING/SPONSOR
   (WRD (MEETING CONFERENCE WORKSHOP SYMPOSIUM)
        (SETRQ SEMHEAD DB/CONFERENCE)
        (SETR HEAD (BUILDQ (N G)
                           (ROOTF * N)))
        (SETRQ NU SG)
        (TO MEETING/MEETING)))


(MEETING/THE
   (WRD (NEXT LAST UPCOMING)
        (PBDRY)
        (SETRQ TIME T)
        (SELECTQ LEX (UPCOMING (SETRQ SEMQUANT AIL)
                               (ADDL SEMLINKS
                                     (QUOTE (TIME (AFTER NOW)))))
                 (NEXT (SETRQ SEMQUANT (ORDINAL NEXT)))
                 (LAST (SETRQ SEMQUANT (ORDINAL LAST)))
                 NIL)
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO MEETING/MOD))
   (JUMP MEETING/MOD
        (SETRQ SEMQUANT THE)))


(NAME/
   (CAT FIRSTNAME
        (PBDRY)
        (SETRQ NAME T)
        (SETR FIRST *)
        (TO NAME/FIRST))
   (CAT LASTNAME
        (PBDRY)
        (SETRQ NAME T)
        (COND ((CATCHECK (WORD *)
                         (QUOTE FIRSTNAME))
               (SETR AMBIG *))
              (T (SETR LAST *)))
        (TO NAME/POP)))


(NAME/FIRST
   (JUMP NAME/POP
        (* TAKE THIS ARC IF NAME IS AMBIG, OTHER LASTNAME ARC WILL HAVE
           GOTTEN IT)
        (SCOPE (NAME/)
               (VERIFY (NOT (CATCHECK (WORD (GETF FIRST))
                                      (QUOTE LASTNAME))))))
   (CAT LASTNAME
        (SCOPE (NAME/)
               (VERIFY (GOODNAME (GETR FIRST)
                                 LEX)))
        (SETR LAST *)
        (TO NAME/POP)))
```



```
(NAME/POP
   (POP (MPRBUILD)
        (SCOPE (NAME/)
               (SETR INTERP (BUILDQ (! THE * PERSON *)
                                    (GENSYM)
                                    (COND ((AND (GETR FIRST)
                                                (GETR LAST))
                                           (BUILDQ ((FIRSTNAME *)
                                                    (LASTNAME *))
                                                   FIRST LAST))
                                          ((GETR FIRST)
                                           (BUILDQ ((FIRSTNAME *))
                                                   FIRST))
                                          ((GETR LAST)
                                           (BUILDQ ((LASTNAME *))
                                                   LAST))))
                                    (LIFTR INTERP))))


(NUM/
   (CAT DIGITS
        (PBDRY)
        (SETR NUM (NUMBOF *))
        (TO NUM/NUM))
   (CAT TEENS
        (PBDRY)
        (SETR NUM (NUMBOF *))
        (TO NUM/NUM))
   (CAT TENS
        (SETR NUM (NUMBOF LEX))
        (TO NUM/1)))


(NUM/1
   (CAT DIGITS
        (SCOPE (NUM/)
               (SETR NUM (IPLUS (NUMBOF *)
                                (GETR NUM))))
        (TO NUM/NUM))
   (JUMP NUM/NUM ))


(NUM/NUM
   (POP (GETR NUM) ))


(NUMBER/
   (PUSH NUM/
        (SETR NUM (NUMBOF *))
        (TO NUMBER/NUM))
   (WRD A
        (TO NUMBER/A)))
```

```
(NUMBER/A
   (WRD HUNDRED
        (PBDRY)
        (SETRQ NUM 100)
        (TO NUMBER/HUNDRED))
   (WRD THOUSAND
        (PBDRY)
        (SETRQ NUM 1000)
        (TO NUMBER/THOUSAND)))


(NUMBER/AND
   (POSN NUM/
        (SCOPE (NUMBER/)
               (SETR NUM (IPLUS (GETR NUM)
                                (NUMBOF *))))
        (TO NUMBER/POP)))


(NUMBER/HUNDRED
   (JUMP NUMBER/POP )
   (WRD AND
        (TO NUMBER/AND))
   (JUMP NUMBER/AND ))


(NUMBER/NUM
   (POP (GETR NUM)
        (VERIFY (NOT (WRD (HUNDRED THOUSAND)))))
   (WRD HUNDRED
        (SCOPE (NUMBER/)
               (SETR NUM (ITIMES 100 (GETR NUM))))
        (TO NUMBER/HUNDRED))
   (WRD THOUSAND
        (SCOPE (NUMBER/)
               (SETR NUM (ITIMES (GETR NUM)
                                 1000)))
        (TO NUMBER/THOUSAND)))


(NUMBER/POP
   (POP (GETR NUM)
        (VERIFY (NOT (WRD (HUNDRED THOUSAND))))))


(NUMBER/THOUSAND
   (JUMP NUMBER/POP )
   (WRD AND
        (TO NUMBER/AND))
   (CAT DIGITS
        (SCOPE (NUMBER/)
               (SETR NUM (IPLUS (GETR NUM)
                                (ITIMES 100 (NUMBOF *)))))
        (TO NUMBER/THOUSAND-NUM)))
```



```
(NUMBER/THOUSAND-NUM
   (WRD HUNDRED
        (TO NUMBER/HUNDRED))
   (JUMP NUMBER/POP ))


(ORD/
   (CAT ORD
        (PBDRY)
        (SETR ORD (NUMBOF *))
        (TO ORD/ORD))
   (CAT TENS
        (PBDRY)
        (* "THIRTY FIFTH")
        (SETR ORD (NUMBOF *))
        (TO ORD/NUM)))


(ORD/NUM
   (CAT ORD
        (VERIFY (AND (NUMBERP (NUMBOF *))
                     (ILESSP (NUMBOF *)
                             10)))
        (SCOPE (ORD/)
               (SETR ORD (IPLUS (NUMBOF *)
                                (GETR ORD))))
        (TO ORD/ORD)))


(ORD/ORD
   (POP (GETR ORD) ))


(P-DETQ-PL/
   (WRD HOW
        (TO P-DETQ/MANY?))
   (WRD (WHAT WHICH)
        (SETRQ SEQUANT EVERY)
        (SETR DET (BUILDQ (DET *)
                          LET))
        (TO PEOPLE/NUM)))


(P-DETQ/MANY?
   (WRD MANY
        (SETRQ SEQUANT HOW@MANY)
        (SETRQ DET (DET HOW@MANY))
        (TO PEOPLE/NUM)))
```

```
(PEOPLE-PL/
   (JUMP PEOPLE/NUM
      (SETRQ SEMQUANT SOME))
   (CAT INTEGER
      (PBDRY)
      (VERIFY (NOT (FMEMB LEX '(QUOTE (ZERO OH ONE)))))
      (SETR SEMQUANT (NUMBOF LEX))
      (SETR DET (BUILDQ (DET (QUANT (INTEGER *)))))
      (TO PEOPLE/NUM)))


(PEOPLE/
   (JUMP PEOPLE-PL/ )
   (JUMP PERSON/ ))


(PEOPLE/NUM
   (WRD PEOPLE
      (SCOPE (PEOPLE-PL/)
             (AND (NULLR DET)
                  (PBDRY)))
      (SETRQ SEMHEAD PERSON)
      (SETRQ N PERSON)
      (SETRQ NU PL)
      (SETRQ HEAD (N PERSON))
      (TO PEOPLE/P)))


(PEOPLE/P
   (JUMP PEOPLE/POP ))


(PEOPLE/POP
   (POP (NPBUILD)
      (SCOPE (PEOPLE/ P-DETQ-PL/ PEOPLE/NUM PECPLE-PL/ PERSON/)
             (COND ((NULLR INTERP)
                    (SETR INTERP (BUILDQ (! * 1 * *)
                                         SEMQUANT
                                         (GENSYM)
                                         SEMHEAD SEMLINKS))))
             (LIFTR INTERP))))
```



```
(PERSON/
   (WRD (A ONE)
      (SETRQ SEMQUANT 1)
      (SETRQ DET (DET (ART A)))
      (TO PERSON/A))
   (WRD (SOMEONE ANYONE ANYBODY SOMEBODY EVERYONE EVERYBODY)
      (SETRQ SEMQUANT (SELECTQ LEX ((SOMEONE ANYONE ANYBODY SOMEBODY)
                                    (QUOTE SOME))
                                   (QUOTE ALL)))
      (SETRQ DET (DET))
      (SETR N (BUILDQ (PRO *)))
      (SETRQ NU SG)
      (SETR HEAD (BUILDQ (PRO *)))
      (TO PEOPLE/P))
   (PUSH NAME/
      (SETR INTERP (GETF INTERP))
      (SETR N (UNHASH *))
      (TO PERSON/POP)))


(PERSON/A
   (WRD PERSON
      (SETRQ SEMHEAD PERSON)
      (SETR N *)
      (SETRQ NU SG)
      (SETR HEAD (BUILDQ (N *)))
      (TO PEOPLE/P)))


(PERSON/POP
   (POP (GETR N)
      (LIFTR INTERP)))


(PP/POP
   (POP (GETR PP)
      (LIFTR SEMLINKS)))
```

```
; <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:38

(PROJECT/
   (CAT PROJECT
        (ADDR PROJNAME *)
        (COND ((NOLLR NAMES)
               (SETR NAMES (ASKFORF (QUOTE TRIP)
                                    (LIST (QUOTE PROJECTNAMES)
                                          (KWOTE LEX))))
               (SETR NAMELIST (GETR NAMES))
               (PBDRY)))
        (OR (NULL (GETR NAMES))
            (VERIFY (SETR NAMES (VALIDPROJS (GETR PROJNAME)
                                            (GETR NAMES)))))
        (TO PROJECT/))
   (POP (GETR PROJNAME)
        (SCOPE (PROJECT/)
               (SETR INTERP (GETR PROJNAME))
               (LIFTR INTERP))
        (VERIFY (WHOLEPROJ? (GETR PROJNAME)))))


(S/
   (JUMP START/ )
   (WRD -PAUSE-
        (TO START/)))


(S/ADD
   (PUSH ARITH/
        (SETR SEMNUM *)
        (SETR OBJ *)
        (TO S/ADD-X))
   (WRD IT
        (SETRQ SEMNUM (NUMINFOCUS))
        (SETRQ OBJ (NP (PRO IT)
                       FEATS (NU SG))))
        (TO S/ADD-X)))


(S/ADD-X
   (WRD (TO AND)
        (TO S/ADD-X-TO)))


(S/ADD-X-TO
   (PUSH ARITH/
        (SCOPE (S/IMP)
               (SETR INTERP (BUILDQ (+ + *)
                                    SEMARITH SEMNUM)))
        (SETR INDOBJ *)
        (TO S/POP)))
```

```
; <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:39

(S/AM
   (WRD I
        (SETRQ SEMQUANT ANY)
        (INCORP SEMLINKS (QUOTE (TRAVELER SPEAKER)))
        (SETR SUBJ (BUILDQ (NP (PRO I)
                               (FEATS (NU SG))))
        (TO S/IS-PERSON)))


(S/ARE
   (WRD WE
        (SETRQ SUBJ (NP (PRO WE)
                        (FEATS (NU PL))))
        (TO S/ARE-WE)))


(S/ARE-WE
   (WRD (UNDER-BUDGET-ED OVER-BUDGET-ED)
        (SETR INTERP (BUILDQ (! THE * DB/BUDGET ((BUDGET/FOR SPEAKER)))
                             (GENSYM)))
        (SETR ACTION (BUILDQ (TEST: (# ***))
                             (COND ((EQ LEX (QUOTE UNDER-BUDGET-ED))
                                    (QUOTE UNDERBUDGET?))
                                   (T (QUOTE OVERBUDGET?)))))
        (SCOPE (S/)
               (SETR V (ROOTP * V)))
        (TO S/OVER-BUDGET))
   (WRD (OVER UNDER)
        (SETR INTERP (BUILDQ (! THE * DB/BUDGET ((BUDGET/FOR SPEAKER)))
                             (GENSYM)))
        (SETR ACTION (BUILDQ (TEST: (# ***))
                             (COND ((EQ LEX (QUOTE OVER))
                                    (QUOTE OVERBUDGET?))
                                   (T (QUOTE UNDERBUDGET?)))))
        (SCOPE (S/)
               (SETR V *))
        (TO S/ARE-WE-OVER)))


(S/ARE-WE-OVER
   (WRD (THE OUR)
        (SELECTQ LEX (THE (SETRQ DET (DET THE)))
                 (OUR (SETRQ DET (DET (POSS OUR))))
                 NIL)
        (TO S/ARE-WE-OVER-THE)))


(S/ARE-WE-OVER-THE
   (WRD BUDGET
        (SCOPE (S/ARE-WE-OVER)
               (SETR OBJ (BUILDQ (NP + (N *)
                                      (FEATS (NU SG)))
                                 DET)))
        (TO S/OVER-BUDGET)),
```

```
(S/ARITH
   (PUSH ARITH/
        (SCOPE (S/WHAT-IS-THE)
               (SETR INTERP (BUILDQ (* * *)
                                    SEMARITH SEMNUM)))
        (SCOPE (S/WHAT-IS)
               (SETR SUBJ (BUILDQ (M * * *)
                                  SEMARITH SUBJ)))
        (TO S/POP)))


(S/ARRANGE
   (PUSH T-DET-SG-INDEF/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR OBJ *)
        (TO S/POP))
   (PUSH T-PL/DET
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR OBJ *)
        (TO S/POP))
   (WRD FOR
        (TO S/ARRANGE-FOR)))


(S/ARRANGE-FOR
   (PUSH PERSON/
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (GETP INTERP)))
        (SETR SUBJX *)
        (TO S/ARRANGE-FOR-PERSON)))


(S/ARRANGE-FOR-PERSON
   (WRD TO
        (TO S/ARRANGE-FOR-PERSON-TO)))


(S/ARRANGE-FOR-PERSON-TO
   (PUSH TO/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SCOPE (S/ARRANGE-FOR)
               (SETR OBJ (REPLACESOM SUBJ * (GETR SUBJX))))
        (TO S/POP)))
```



```
(S/ASSUME-BUDGET-HAS
   (PUSH AMOUNT/
        (SETR ACTION (BUILDQ (ADD: *** MONEY/ALLOCATED #)
                             (GETP AMT)))
        (SETR OBJ (BUILDQ (NP (S (THAT (SUBJ *)
                                       (AUX (TNS PRESENT)
                                            (VOICE ACITVE))
                                       (VP (V HAVE)
                                           (NP *)))))
                          SUBJX))
        (TO S/POP)))


(S/ATTEND
   (PUSH MEETING/
        (INCORP SEMLINKS (BUILDQ (TO/ATTEND #)
                                 (GETP INTERP)))
        (SETR OBJ *)
        (TO S/POP)))


(S/BUDGET-HAS
   (PUSH AMOUNT/
        (SCOPE (S/)
               (COND ((EQ (GETR TYPE)
                          (QUOTE DCL))
                      (SETR ACTION (BUILDQ (ADD: *** MONEY/ALLOCATED #)
                                           (GETP AMT))))
                     (T (ADDL SEMLINKS (BUILDQ (MONEY/ALLOCATED #)
                                               (GETP AMT))))))
        (* CURRENTLY, ALL MODIFIERS ON AMOUNTS ARE IGNORED SEMANTICALLY)
        (SETR OBJ *)
        (TO S/POP))
   (WRD A
        (SCOPE (S/)
               (VERIFY (NOT (EQ (GETR TYPE)
                                (QUOTE DCL)))))
        (TO S/BUDGET-HAS-A)))


(S/BUDGET-HAS-A
   (WRD (SURPLUS DEFICIT)
        (SCOPE (S/)
               (SETR SEMTYPE NIL))
        (SELECTQ LEX
                 (SURPLUS (SETRQ ACTION
                                 (TEST: (GREATERP (GET: ***
                                                        MONEY/REMAINING)
                                                  0))))
                 (DEFICIT (SETRQ ACTION
                                 (TEST: (LESSP (GET: *** MONEY/REMAINING)
                                               0))))
                 NIL)
        (SETR OBJ (BUILDQ (NP (N *)
                              (FEATS (NO SG)))))
        (TO S/POP)))
```

```
(S/CANCEL
   (PUSH T-DET-DEF-OR-SG/
       (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                (CDR (GETF INTERP)))
       (SETR OBJ *)
       (TO S/POP)))


(S/CHARGE
   (PUSH AMOUNT/
       (SETR SEMTEMP (GETF INTERP))
       (SETR OBJ *)
       (TO S/CHARGE-Y;)
   (PUSH EXPENSE-DET/
       (SETR SEMTEMP (GETF INTERP))
       (SETR OBJ *)
       (TO S/CHARGE-X))
   (PUSH T-DET-DEF-OR-SG/
       (SETR SEMTEMP (GETF INTERP))
       (SETR OBJ *)
       (TO S/CHARGE-X)))


(S/CHARGE-X
   (WRD TO
       (TO S/CHARGE-X-TO)))


(S/CHARGE-X-TO
   (PUSH BUDGET/
       (SETR INTERP (GETF INTERP))
       (SETR ACTION (BUILDQ (CHARGE: * ***)
                            SEMTEMP))
       (ADDL VMODS (BUILDQ (PP (PREP TO)
                               *)))
       (TO S/POP))
   (PUSH BUDGET-ITEM/
       (SETR INTERP (GETF INTERP))
       (SETR ACTION (BUILDQ (CHARGE: * ***)
                            SEMTEMP))
       (ADDL VMODS (BUILDQ (PP (PREP TO)
                               *)))
       (TO S/POP)))


(S/COMPUTE
   (PUSH EXPENSE-DET/
       (SETR OBJ *)
       (TO S/POP)))
```



```
(S/DCL-IS
   (JUMP S/DCL-IS-ADV )
   (WRD -PAUSE-
       (TO S/DCL-IS-ADV)))


(S/DCL-IS-ADV
   (PUSH AMOUNT/
       (SETR ACTION (BUILDQ (PROGN (ADD: *** VALUE 0)
                                   (ADD: *** UNITS DOLLARS))
                            (GETF AMT)))
       (SETR OBJ (BUILDQ (MP *)))
       (TO S/POP)))


(S/DCL-SUBJ-PL
   (WRD (ARE WERE COST COST-PAST)
       (COND ((PNEMB LEX (QUOTE (COST COST-PAST)))
              (PBDRY)))
       (SETR V (ROOTF LEX V))
       (SETR HEAD (GETR V))
       (SETR TNS (CATF LEX V TNS))
       (SETRQ VOICE ACTIVE)
       (INCORP SEMLINKS (TIMEOF (GETR TNS)))
       (TO S/DCL-IS)))


(S/DCL-SUBJ-SG
   (WRD (IS WAS COSTS COST-PAST)
       (COND ((PNEMB LEX (QUOTE (COSTS COST-PAST)))
              (PBDRY)))
       (SETR V (ROOTF LEX V))
       (SETR HEAD (GETR V))
       (SETR TNS (CATF LEX V TNS))
       (SETRQ VOICE ACTIVE)
       (INCORP SEMLINKS (TIMEOF (GETR TNS)))
       (TO S/DCL-IS)))


(S/DID
   (PUSH PEOPLE/
       (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                              (GETF INTERP)))
       (SETR SUBJ *)
       (TO S/DID-PERSON)))
```

```
(S/DID-PERSON
   (WRD ATTEND
        (SETR V (ROOTP LEX V))
        (SETR HEAD (GETR V))
        (TO S/ATTEND))
   (WRD (GO FLY TRAVEL)
        (AND (EQ LEX (QUOTE FLY))
             (ADDL SEMLINKS (BUILDQ (MODE/CP/TRANSPORT AIR))))
        (SETR V (ROOTP LEX V))
        (SETR HEAD (ROOTP LEX V))
        (TO S/PERSON-GO)))


(S/DO
   (WRD (I WE)
        (SETRQ SEMQUANT THE)
        (SETRQ SEMHEAD DB/BUDGET)
        (ADDL SEMLINKS (BUILDQ (BUDGET/OF (! SOME # DB/CONTRACT
                                             ((GROUP #))))
                               (GENSYM)
                               (BUILDQ (! SOME # DB/GROUP
                                          ((MEMBERS #)))
                                       (GENSYM)
                                       (SEMPRO LEX))))
        (SETR SUBJ (BUILDQ (NP (PRO YOU)
                               (FEATS (NU SG)))))
        (TO S/DOES-BUDGET)))


(S/DOES
   (PUSH BUDGET/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/DOES-BUDGET)))


(S/DOES-BUDGET
   (WRD HAVE
        (SCOPE (START/)
               (SETRQ V HAVE))
        (TO S/BUDGET-HAS)))
```



```
(S/ENTER
   (PUSH T-ENTER/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETP OBJ *)
        (TO S/POP))
   (PUSH BUDGET/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR OBJ *)
        (TO S/POP))
   (PUSH BUDGET-ITEM/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETP OBJ *)
        (TO S/POP)))


(S/ESTIMATE
   (PUSH EXPENSE-DET/
        (SETR INTERP (GETP INTERP))
        (SETR OBJ *)
        (TO S/POP)))


(S/ESTIMATE-THE
   (PUSH EXPENSE-DET/
        (SCOPE (S/ESTIMATE)
               (SETR OBJ (DETCOMBINE (GETR DET)
                                     (UNHASH *))))
        (TO S/POP)))


(S/HOW
   (WRD LONG
        (SETRQ ACTION (OUTPUT: (GET: *** DURATION)))
        (ADDL VMODS (QUOTE (ADV HOW@LONG)))
        (TO S/HOW-LONG))
   (WRD MUCH
        (SETRQ ACTION (OUTPUT: (GET: *** COST)))
        (TO S/HOWMUCH)))


(S/HOW-LONG
   (WRD (IS WAS)
        (SETRQ VOICE ACTIVE)
        (SETR HEAD *)
        (SETR V *)
        (SETR TNS (CATP LEX V TNS))
        (SETR SEMTNS (TIMEOF (GETR TNS)))
        (TO S/HOWLONG-IS))
   (WRD WILL
        (SETR SEMTNS (QUOTE (TIME (AFTER NOW))))
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS FUTURE)
        (TO S/HOWLONG-WILL)))
```

```
(S/HOWLONG-IS
   (PUSH T-DET-SG-DEF/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (INCORP SEMLINKS (GETR SEMTNS))
        (SETR SUBJ *)
        (TO S/POP))
   (PUSH MEETING/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (INCORP SEMLINKS (GETR SEMTNS))
        (SETR SUBJ *)
        (TO S/POP)))


(S/HOWLONG-WILL
   (PUSH T-DET-SG-DEF/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SCOPE (S/HOWLONG)
               (INCORP SEMLINKS (GETR SEMTNS)))
        (SETR SUBJ *)
        (TO S/HOWLONG-WILL-TRIP))
   (PUSH MEETING/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SCOPE (S/HOWLONG)
               (INCORP SEMLINKS (GETR SEMTNS)))
        (SETR SUBJ *)
        (TO S/HOWLONG-WILL-TRIP)))


(S/HOWLONG-WILL-TRIP
   (WRD (LAST BE)
        (AND (EQ LEX (QUOTE LAST))
             (PBDRY))
        (SETRQ V BE)
        (SETRQ HEAD BE)
        (TO S/POP)))
```



```
(S/HOWMANY-PEOPLE
   (WRD (ARE WERE)
        (SETR TNS (CATP LEX V TNS))
        (INCORP SEMLINKS (TIMEOF (GETR TNS)))
        (TO S/IS-PERSON))
   (WRD (WENT TRAVEL-ED FLEW)
        (AND (EQ LEX (QUOTE FLEW))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETRQ TNS PAST)
        (SETR V (ROOTP LEX V))
        (SETR HEAD (GETR V))
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-GO))
   (WRD WILL
        (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
        (SETRQ TNS FUTURE)
        (TO S/PERSON-WILL))
   (WRD ATTEND-ED
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (SETRQ TNS PAST)
        (SETRQ V ATTEND)
        (SETRQ HEAD ATTEND)
        (SETRQ VOICE ACTIVE)
        (TO S/ATTEND)))


(S/HOWMANY-TRIPS
   (WRD (REMAIN ARE)
        (AND (EQ LEX (QUOTE REMAIN))
             (PBDRY))
        (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS PRES)
        (SETR V *)
        (SETR HEAD *)
        (TO S/HOWMANY-TRIPS-REMAIN))
   (WRD ARE
        (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
        (TO S/HOWMANY-TRIPS-ARE))
   (WRD (DID WILL)
        (SETR SEMTNS (TIMEOF (GETR TNS)))
        (SETRQ VOICE ACTIVE)
        (SETR TNS (CATP LEX MODAL TNS))
        (TO S/HOWMANY-TRIPS-WILL))
   (WRD HAS
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/HOWMANY-TRIPS-HAS))
```

```
   (WRD WERE
        (SETRQ TNS PAST)
        (SETRQ VOICE PASSIVE)
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ (BUILDQ (NP (PRO SOMEONE)
                                      (FEATS (NU SG))))))
        (TO S/HOWMANY-TRIPS-HAS-PERSON))
   (WRD HAVE
        (SETRQ TNS PAST)
        (TO S/HOWMANY-TRIPS-HAVE)))


(S/HOWMANY-TRIPS-ARE
   (WRD THERE
        (SETRQ V BE)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (TO S/POP)))


(S/HOWMANY-TRIPS-HAS
   (PUSH PERSON/
        (SETR SEMSUBJ (GETF INTERP))
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ *);
        (TO S/HOWMANY-TRIPS-HAS-PERSON))
   (WRD (HE SHE)
        (SETR SEMSUBJ (SEMPRO LEX))
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ (BUILDQ (NP (PRO *)
                                      (FEATS (NU SG))))))
        (TO S/HOWMANY-TRIPS-HAS-PERSON)))
```



```
(S/HOWMANY-TRIPS-HAS-PERSON
   (WRD TAKEN
        (PBDRY)
        (SCOPE (S/HOWMANY-TRIPS-HAS)
               (INCORP SEMLINKS (BUILDQ (TRAVELER *)
                                        SEMSUBJ)))
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (SETP V (ROOTP LEX V))
        (SETR HEAD (ROOTP LEX V))
        (TO S/HOWMANY-TRIPS-TAKEN))
   (WRD (PLAN-ED SCHEDULE-ED ARRANGE-ED)
        (PBDRY)
        (INCORP SEMLINKS (BUILDQ (DB/CREATOR *)
                                 SEMSUBJ))
        (INCORP SEMLINKS (QUOTE (CREATE/TIME (BEFORE NOW))))
        (SETRQ SCHEDULE T)
        (SETR V (ROOTP LEX V))
        (SETR HEAD (GETR V))
        (TO S/HOWMANY-TRIPS-TAKEN))
   (WRD CANCEL-ED
        (PBDRY)
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (INCORP SEMLINKS (QUOTE (MODALITY CANCELED)))
        (SETRQ V CANCEL)
        (SETRQ HEAD CANCEL)
        (TO S/HOWMANY-TRIPS-TAKEN)))


(S/HOWMANY-TRIPS-HAVE
   (WRD BEEN
        (SETRQ SEMSUBJ SOMEONE)
        (SETRQ VOICE PASSIVE)
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ (BUILDQ (NP (PRO SOMEONE)
                                      (FEATS (NU SG))))))
        (TO S/HOWMANY-TRIPS-HAS-PERSON))
   (WRD (WE THEY I)
        (SETR SEMSUBJ (SEMPRO LEX))
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ (BUILDQ (NP (PRO *)
                                      (FEATS (NU 0)))
                                  (CATP LEX PRO NUMBER))))
        (TO S/HOWMANY-TRIPS-HAS-PERSON)))


(S/HOWMANY-TRIPS-REMAIN
   (WRD IN
        (TO S/TRIPS-REMAIN-IN)))
```

```
(S/HOWMANY-TRIPS-TAKEN
   (PUSH DATAMOD/
        (COND ((GETP SCHEDULE)
               (INCORP SEMLINKS (BUILDQ (@ (CREATE/TIME)
                                           #)
                                        (CUR (GETP INTERP)))))
              (T (INCORP SEMLINKS (GETP INTERP))))
        (SCOPE (S/HOWMANY-TRIPS)
               (VERIFY (TIMESENSE (GETR TNS)
                                  *)))
        (ADJL VMODS *)
        (TO S/POP))
   (JUMP S/POP ))


(S/HOWMANY-TRIPS-WILL
   (PUSH PERSON/
        (SETR SEMSUBJ (GETP INTERP))
        (SCOPE (S/)
               (SETR OBJ (GETR SUBJ))
               (SETR SUBJ *))
        (TO S/HOWMANY-TRIPS-WILL-PERSON))
   (CAT PRO-SUBJ
        (SETR SEMSUBJ (SEMPRO LEX))
        (SCOPE (S/)
               (SETR OBJ (GETR SDBJ))
               (SETR SDBJ (BUILDQ (NP (PRO *)
                                      (FEATS (NU #)))
                                  (CATP LEX PRO NUMBER))))
        (TO S/HOWMANY-TRIPS-WILL-PERSON)))


(S/HOWMANY-TRIPS-WILL-PERSON
   (WRD (TAKE MAKE)
        (PBDRY)
        (SCOPE (S/HOWMANY-TRIPS)
               (INCORP SEMLINKS (BDILDQ (TRAVELER *)
                                        SEMSDBJ))
               (INCORP SEMLINKS (GETR SEMTNS)))
        (SETR V *)
        (SETR HEAD *)
        (TO S/HOWMANY-TRIPS-TAKEN))
   (WRD (PLAN SCHEDULE)
        (PBDRY)
        (SCOPE (S/HOWMANY-TRIPS)
               (VERIFY (NOT (EQ (GETR TNS)
                                (QUOTE FDTURE))))
               (INCORP SEMLINKS (BUILDQ (DE/CREATOR *)
                                        SEMSDBJ))
               (INCORP SEMLINKS (BUILDQ (CREATE/TIME #)
                                        (CADR (GETR SEMTNS)))))
        (SETRQ SCHEDULE T)
        (SETR V *)
        (SETR HEAD *)
        (TO S/HOWMANY-TRIPS-TAKEN)))
```



```
(S/HOWMUCH
   (WRD MONEY
        (SETR OBJ (BDILDQ (NP (UET (QDANT HOWMUCH))
                              (N MONEY)
                              (FEATS (NU MASS)))))
        (TO S/HOWMUCH-MONEY))
   (JUMP S/HOWMUCH-MONEY
        (SETR OBJ (BDILDQ (NP (UET (QUANT HOWMUCH))
                              (N ?)
                              (FEATS (NU MASS)))))))


(S/HOWMUCH-IS-IN
   (PUSH BUDGET/
        (SETR INTERP (GETP INTERP))
        (SCOPE (S/)
               (SETRQ ACTION (OUTPUT: (GET: *** MONEY/REMAINING))))
        (ADJL VMODS (BUILDQ (PP (PREP IN)
                                *)))
        (TO S/POP)))


(S/HOWMUCH-MONEY
   (WRD (DOES DID WILL WOULD)
        (SETR TNS (CATP LEX MODAL TNS))
        (SETRQ VOICE ACTIVE)
        (SETR SEMTNS (TIMEOF (GETR TNS)))
        (TO S/HOWMUCH-WILL))
   (WRD (IS WAS)
        (SETRQ V BE)
        (SETRQ VOICE ACTIVE)
        (SETRQ HEAD BE)
        (SETR TNS (CATP LEX V TNS))
        (SETR PNCODE (CATP LEX V PNCODE))
        (SETR SEMTNS (TIMEOF (GETR TNS)))
        (TO S/WHAT-IS))
   (WRD (DO DID)
        (SETR TNS (CATP LEX V TNS))
        (SETR SEMTNS (TIMEOF (GETR TNS)))
        (TO S/WHAT-DO))
   (WRD HAVE
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS PAST)
        (SETRQ ASPECT PERFECT)
        (TO S/WHAT-HAVE)))


(S/HOWMUCH-WE-SPEND
   (PUSH DATAMOD-EXTENT/
        (INCORP SEMLINKS (GETP INTERP))
        (ADJL VMODS *)
        (TO S/POP))
   (JUMP S/POP ))
```

```
(S/HOWMUCH-WILL
   (CAT PRO-SUBJ
        (ADDL SEMLINKS (BUILDQ (BUDGET/FOR #)
                               (SEMPRO LEX)))
        (ADDL SEMLINKS (GETR SEMTNS))
        (SETRQ SEMHEAD DB/BUDGET)
        (VERIFY (NOT (EQ LEX (QUOTE YOU))))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU #)))
                           (CATP LEX PRO NUMBER)))
        (TO S/HOWMUCH-WILL-WE))
   (PUSH PERSON/
        (ADDL SEMLINKS (BUILDQ (BUDGET/FOR #)
                               (GETP INTERP)))
        (ADDL SEMLINKS (GETR SEMTNS))
        (SETRQ SEMHEAD DB/BUDGET)
        (SETR SUBJ (BUILDQ *))
        (TO S/HOWMUCH-WILL-WE)))


(S/HOWMUCH-WILL-WE
   (WRD SPEND
        (PBODY)
        (SETR V *)
        (SETR HEAD *)
        (TO S/HOWMUCH-WE-SPEND)))


(S/I
   (WRD (WANT NEED)
        (SETR V *)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (TO S/I-WANT)))


(S/I-WANT
   (WRD TO
        (INCORP SEMLINKS (QUOTE (TRAVELER SPEAKER)))
        (TO S/I-WANT-TO))
   (WRD (AN A)
        (SETRQ V LIST)
        (SETRQ ARTAGREE T)
        (SETRQ SUBJ (NP (PRO YOU)
                        (FEATS (NU SG))))
        (TO S/SHOW-ME-A)))


(S/I-WANT-TO
   (PUSH TO/
        (SCOPE (S/)
               (SETR OBJ (REPLACESON SUBJ * (GETR SUBJ))))
        (TO S/POP)))
```



```
(S/IMP
   (WRD ADD
        (SETRQ SEMARITH PLUS)
        (SETR V *)
        (SETR HEAD *)
        (TO S/ADD))
   (WRD (ADD ENTER CREATE MAKE)
        (SETRQ SEMFORM (BUILDQ (@ (BUILDQ: #)
                                  #)
                               (SEMHEAD INTERP)
                               (SEMLINKS INTERP)))
        (SETR V *)
        (SETR HEAD *)
        (TO S/ENTER))
   (WRD (SCHEDULE PLAN)
        (SETRQ SEMFORM (BUILDQ (@ (BUILDQ: #)
                                  #)
                               (SEMHEAD INTERP)
                               (SEMLINKS INTERP)))
        (SETR V *)
        (SETR HEAD *)
        (TO S/PLAN))
   (WRD (CANCEL FORGET DELETE DISREGARD)
        (SETRQ ACTION (CANCEL; ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/CANCEL))
   (WRD (SHOW GIVE)
        (SETRQ ACTION (OUTPUT: ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/SHOW))
   (WRD LIST
        (SETRQ ACTION (OUTPUT: ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/SHOW-ME-LIST))
   (WRD PRINT
        (SETRQ ACTION (OUTPUT: ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/PRINT))
   (WRD ARRANGE
        (SETRQ SEMFORM (BUILDQ (@ (BUILDQ: #)
                                  #)
                               (SEMHEAD INTERP)
                               (SEMLINKS INTERP)))
        (SETR V *)
        (SETR HEAD *)
        (TO S/ARRANGE))
```

```
   (WRD CHARGE
        (SETR V *)
        (SETR HEAD *)
        (TO S/CHARGE))
   (WRD (FIND COMPUTE FIGURE ESTIMATE-V CALCULATE ITEMIZE)
        (SETRQ ACTION (OUTPUT: ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/ESTIMATE))
   (WRD (DISPLAY GET)
        (SETRQ ACTION (OUTPUT: ***))
        (SETR V *)
        (SETR HEAD *)
        (TO S/SHOW-ME)))


(S/IS
   (PUSH PERSON/
        (INCORP SEMLINKS (BUILDQ (TRAVELER *)
                                 (GETP INTERP)))
        (SETP SUBJ *)
        (TO S/IS-PERSON))
   (WRD (HE SHE)
        (INCORP SEMLINKS (BUILDQ (TRAVELER *)
                                 (SEMPRO LEX)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU *)))
                           (CATP LEX PRO NUMBER)))
        (TO S/IS-PERSON)))


(S/IS-PERSON
   (WRD ATTEND-ING
        (SCOPE (S/)
               (SETR HEAD (ROOTP LEX V))
               (SETR V (GETR HEAD))
               (COND ((GETR TNS)
                      (SETR TNS (CONS (GETR TNS)
                                      (QUOTE (PROGRESSIVE)))))
                     (T (SETRQ TNS (PRESENT PROGRESSIVE)))))
        (SETRQ VOICE ACTIVE)
        (TO S/ATTEND))
   (WRD (GO-ING FLY-ING TRAVEL-ING)
        (AND (EQ LEX (QUOTE FLY-ING))
             (ADDL SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SCOPE (S/)
               (SETR HEAD (ROOTP LEX V))
               (SETR V (ROOTP LEX V))
               (COND ((GETR TNS)
                      (SETR TNS (CONS (GETR TNS)
                                      (QUOTE (PROGRESSIVE)))))
                     (T (SETRQ TNS (PRESENT PROGRESSIVE)))))
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-GO))
```



```
   (WRD SCHEDULE-ED
        (SCOPE (S/)
               (SETRQ HEAD SCHEDULE)
               (SETRQ V SCHEDULE)
               (COND ((GETR TNS)
                      (SETRQ VOICE PASSIVE))
                     (T (SETRQ VOICE ACTIVE)
                        (SETRQ TNS PRESENT))))
        (TO S/IS-PERSON-SCHEDULED)))


(S/IS-PERSON-SCHEDULED
   (WRD TO
        (TO S/IS-PERSON-SCHEDULED-TO)))


(S/IS-PERSON-SCHEDULED-TO
   (PUSH TO/
        (MERGE SEMLINKS (GETP SEMLINKS))
        (SCOPE (S/)
               (SETR OBJ (REPLACESOM SUBJ * (GETR SUBJ)))
               (SETRQ SUBJ (NP (PRO SOMEONE)
                               (FEATS (NU SG)))))
        (TO S/POP)))


(S/MILEAGE
   (WRD (ON FOR)
        (SETR PREP *)
        (TO S/PURPOSE-OF)))


(S/OVER-BUDGET
   (PUSH DATEMOD-EXTENT/
        (SCOPE (S/)
               (VERIFY (TIMESENSE (GETR TNS)
                                  *)))
        (INCORP SEMLINKS (GETP INTERP))
        (ADDL VMODS *)
        (TO S/POP))
   (JUMP S/POP ))


(S/PEOPLE
   (JUMP S/HOWMANY-PEOPLE )
   (WRD WILL
        (INCORP SEMLINKS (BUILDQ (TIME (AFTER NOW))))
        (SETRQ TNS FUTURE)
        (TO S/PERSON-WILL))
   (WRD WANT
        (PBDRY)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (SETR HEAD *)
        (SETR V *)
        (TO S/I-WANT)))
```

```
(S/PERSON
   (WRD IS
        (SCOPE (S/)
               (VERIFY (NOT (FIND (QUOTE I)
                                  (GETR SUBJ)))))
        (SETRQ TNS PRESENT)
        (TO S/IS-PERSON))
   (WRD (WENT TRAVEL-ED FLEW)
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (AND (EQ LEX (QUOTE FLEW))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR HEAD (ROOTP LEX V))
        (SETR V (GETR HEAD))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-GO))
   (WRD WILL
        (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
        (SETRQ TNS FUTURE)
        (TO S/PERSON-WILL))
   (WRD ATTEND-ED
        (INCORP SEMLINKS (QUOTE (TIME (BEFOR NOW))))
        (SETR HEAD (ROOTP LEX V))
        (SETR V (GETR HEAD))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/ATTEND))
   (WRD LEFT
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (SETRQ HEAD LEAVE)
        (SETRQ V LEAVE)
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-LEAVE)))


(S/PERSON-GO
   (PUSH TO-PLACE/
        (INCORP SEMLINKS (GETP SEMLINKS))
        (SCOPE (S/)
               (VERIFY (NOT (EQ (GETR QADV)
                                (QUOTE WHERE)))))
        (ADDL VMODS *)
        (TO S/PERSON-GO-MEETING))
   (WRD (ANYWHERE AWAY)
        (SCOPE (S/)
               (VERIFY (NOT (EQ (GETR QADV)
                                (QUOTE WHERE)))))
        (SETR OBJ (BUILDQ (NP (PRO *)
                              (FEATS (NO SG)))))
        (SETR DESTINATION (GETR OBJ))
        (TO S/PERSON-GO-MEETING))
   (JUMP S/PERSON-GO-MEETING
        (SCOPE (START/)
               (VERIFY (GETR WHERE)))))
```



```
(S/PERSON-GO-DATE
   (JUMP S/POP ))


(S/PERSON-GO-MEETING
   (JUMP S/POP )
   (PUSH DATEMOD/
        (INCORP SEMLINKS (GETP INTERP))
        (SCOPE (S/)
               (VERIFY (NOT (EQ (GETR QADV)
                                (QUOTE WHEN))))
               (VERIFY (TIMESENSE (GETP TNS)
                                  *)))
        (ADDL VMODS *)
        (SETR TIME *)
        (TO S/PERSON-GO-DATE)))


(S/PERSON-LEAVE
   (PUSH DATEMOD/
        (INCORP SEMLINKS (GETP INTERP))
        (ADDL VMODS *)
        (TO S/POP)))


(S/PERSON-VISIT
   (PUSH CITY/
        (INCORP SEMLINKS (BUILDQ (DESTINATION *)
                                 (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/PERSON-GO-MEETING))
   (CAT NPR
        (INCORP SEMLINKS (BUILDQ (DESTINATION *)))
        (SETR SUBJ (BUILDQ (NP (N *))))
        (TO S/PERSON-GO-MEETING)))


(S/PERSON-WILL
   (WRD ATTEND
        (PBDRY)
        (SETR V *)
        (SETR HEAD *)
        (SETRQ VOICE ACTIVE)
        (TO S/ATTEND))
   (WRD (GO FLY TRAVEL)
        (AND (EQ LEX (QUOTE *LY))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR V *)
        (SETR HEAD *)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-GO))
```

```
   (WRD BE
        (TO S/IS-PERSON))
   (WRD (LEAVE GO DEPART SPLIT START TRAVEL FLY)
        (AND (EQ LEX (QUOTE FLY))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR V *)
        (SETR HEAD *)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-LEAVE)))


(S/PLAN
   (PUSH T-ENTER/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR OBJ *)
        (TO S/POP)))


(S/POP
   (JUMP END/
        (SCOPE (S/)
               (SETR SEMLINKS (CLEANSEM (GETR SEMLINKS)))
               (SETR INTERP (INTERPBUILD))
               (SETR POPVAL (SBUILD))))
   (WRD -PAUSE-
        (SCOPE (S/)
               (SETR INTERP (INTERPBUILD))
               (SETR POPVAL (SBUILD)))
        (TO END/)))


(S/PRINT
   (WRD OUT
        (TO S/SHOW-ME))
   (JUMP S/SHOW-ME ))


(S/PURPOSE
   (WRD OF
        (SETR PREP *)
        (TO S/PURPOSE-OF)))
```



```
(S/PURPOSE-OF
   (PUSH T-DET-SG-DEF/
        (SCOPE (S/)
               (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST
                                  SEMFORM)
                        (CDR (GETP INTERP)))
               (INCORP SEMLINKS (TIMEOF (GETR TNS))))
        (SCOPE (S/WHAT-IS-THE)
               (SETR OBJ (BUILDQ (NP (DET THE)
                                     (N *)
                                     (PP (PREP *)
                                         *))
                                 N PREP)))
        (TO S/POP)))


(S/PUT
   (PUSH AMOUNT/
        (SETR OBJ *)
        (TO S/PUT-$)))


(S/PUT-$
   (WRD IN
        (TO S/PUT-$-IN)))


(S/PUT-$-IN
   (PUSH BUDGET/
        (ADDR WHODS (BUILDQ (PP (PREP IN)
                                *)))
        (TO S/POP)))


(S/QADV
   (WRD (IS WAS)
        (SETRQ V BE)
        (SETR TNS (CAPP LEX V TNS))
        (SETRQ VOICE ACTIVE)
        (INCORP SEMLINKS (TIMEOF (GETR TNS)))
        (TO S/QADV-IS))
   (WRD (ARE WERE)
        (SETRQ V BE)
        (SETR TNS (CAPP LEX V TNS))
        (SETRQ VOICE ACTIVE)
        (INCORP SEMLINKS (TIMEOF (GETR TNS)))
        (TO S/QADV-WERE))
   (WRD AM
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS PRESENT)
        (INCORP SEMLINKS (TIMEOF (GETR TNS)))
        (TO S/QADV-AM))
```

```
   (WRD WILL
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS (CATP LEX MODAL TNS))
        (INCORP SEMLINKS (TIMEOP (GETR TNS)))
        (TO S/QADV-WILL)))


(S/QADV-AM
   (WRD I
        (INCORP SEMLINKS (QUOTE (TRAVELER SPEAKER)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU SG)))))
        (TO S/IS-PERSON)))


(S/QADV-IS
   (PUSH T-DET-SG-DEF/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/POP))
   (PUSH MEETING/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/POP))
   (PUSH PERSON/
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/IS-PERSON))
   (WRD (HE SHE)
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (SEMPRO LEX)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU SG)))))
        (TO S/IS-PERSON)))


(S/QADV-WERE
   (PUSH T-DET-PL-DEF/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/POP))
   (PUSH PEOPLE-PL/
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/IS-PERSON))
   (WRD (WE THEY)
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (SEMPRO LEX)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU PL)))))
        (TO S/IS-PERSON)))
```



```
(S/QADV-WILL
   (PUSH PEOPLE/
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/PERSON-WILL))
   (CAT PRO-SUBJ
        (INCORP SEMLINKS (BUILDQ (TRAVELER #)
                                 (SEMPRO LEX)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (ND #)))
                           (CATP LEX PRO NUMBER)))
        (TO S/PERSON-WILL)))


(S/SHOW
   (WRD ME
        (TO S/SHOW-ME))
   (WRD (IT THEM)
        (SETR OBJ (BUILDQ (NP (PRO *)
                              (FEATS (NU #)))
                          (CATP * PRO NUMBER)))
        (TO S/SHOW-THING))
   (JUMP S/SHOW-ME ))


(S/SHOW-LIST-AND?
   (JUMP S/POP
        (SCOPE (S/)
               (VERIFY (NULLW BOTH))
               (AND (GETR CONJ)
                    (VERIFY (CONJOINABLE (GETR OBJ))))
               (COND ((GETR CONJ)
                      (SETR OBJ (BUILDQ (# (NP)
                                           # *)
                                        (LIST (GETR CONJ))
                                        OBJ))
                      (SETR INTERP (BUILDQ (# (*)
                                              *)
                                           CONJ INTERP)))
                     (T (SETR OBJ (CAR (GETR OBJ)))
                        (SETR INTERP (CAR (GETR INTERP))))))))


(S/SHOW-ME
   (WRD BOTH
        (SCOPE (S/SHOW-ME)
               (VERIFY (NULLW BOTH)))
        (SETR BOTH *)
        (TO S/SHOW-ME-STUFF))
   (JUMP S/SHOW-ME-STUFF )
   (PUSH BUDGET-ITEM/
        (SETR OBJ *)
        (TO S/POP)))
```

```
(S/SHOW-ME-A
   (WRD (LIST REPORT)
        (?BOTH)
        (TO S/SHOW-ME-A-LIST)))


(S/SHOW-ME-A-LIST
   (WRD OF
        (TO S/SHOW-ME-LIST)))


(S/SHOW-ME-ABOUT
   (PUSH T-DET-DEF-OR-SG/
        (SCOPE (S/SHOW-ME-LIST S/TELL-ME)
               (ADDR OBJ (BUILDQ (NP (N *)
                                     (FEATS (NO SG))
                                     (PP (PREP *)
                                         *))
                                 N PREP)))
        (TO S/SHOW-LIST-AND?)))


(S/SHOW-ME-LIST
   (WRD THEM
        (ADDR OBJ (BUILDQ (NP (PRO THEM)
                              (FEATS (NO PL)))))
        (TO S/SHOW-LIST-AND?))
   (PUSH T-DET-PL/
        (ADDR INTERP (GETP INTERP))
        (SCOPE (S/I-WANT S/SHOW-ME S/IMP)
               (ADDR OBJ *))
        (TO S/SHOW-LIST-AND?))
   (PUSH EX-DET-PL/
        (ADDR INTERP (GETP INTERP))
        (SCOPE (S/I-WANT S/SHOW-ME S/IMP)
               (ADDR OBJ *))
        (TO S/SHOW-LIST-AND?)))
```



```
(S/SHOW-ME-STUFF
   (JUMP S/SHOW-ME-LIST )
   (WRD (A AN)
        (SETEQ ARTAGREE T)
        (SETR DET (BUILDQ (DET (ART *))))
        (TO S/SHOW-ME-A));
   (PUSH T-DET-SG/
        (ADDR INTERP (GETP INTERP))
        (ADDR OBJ *)
        (TO S/SHOW-LIST-AND?))
   (PUSH EX-DET-SG/
        (ADDR INTERP (GETP INTERP))
        (ADDL OBJ *)
        (TO S/SHOW-LIST-AND?))
   (PUSH BUDGET-ITEM-QUART/
        (ADDR INTERP (GETP INTERP))
        (ADDL OBJ *)
        (TO S/SHOW-LIST-AND?)))


(S/SHOW-THING
   (WRD TO
        (TO S/SHOW-THING-TO))
   (JUMP S/POP ))


(S/SHOW-THING-TO
   (WRD ME
        (TO S/POP)))


(S/STATUS
   (WRD OF
        (TO S/STATUS-OF)))


(S/STATUS-OF
   (PUSH BUDGET/
        (ADDR WHODS (BUILDQ (PP (PREP OF)
                                *)))
        (SCOPE (S/WHAT-IS-THE)
               (SETR SUBJ (NPBUILD)))
        (TO S/POP)))


(S/THE
   (WRD TRIP
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO S/THE-TRIP)))
```

```
(S/THE-BUDGET
   (WRD (HAS CONTAIN-S)
        (SETR V *)
        (SETRQ VOICE ACTIVE)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (TO S/BUDGET-HAS)))


(S/THE-TRIP
   (WRD NUMBER
        (SETR HEAD (BUILDQ (N *)))
        (SETRQ NU SG)
        (SCOPE (S/THE)
               (SETR SUBJ (NPBUILD)))
        (TO S/TRIP-NUMBER)))


(S/TRIP-NUMBER
   (WRD (IS WAS)
        (SETRQ V BE)
        (SETR TNS (CATP LEX V TNS))
        (SETR SENSES (TIMEOP (GETR TNS)))
        (TO S/TRIP-NUMBER-IS)))


(S/TRIP-NUMBER-IS
   (PUSH TRIPNUM/
        (SETRQ SEMQUANT THE)
        (SETRQ SEMHEAD UB/TRIP)
        (SETR ACTION (BUILDQ (ADD: *** (TRIP# *))))
        (SETR OBJ *)
        (TO S/POP)))


(S/TRIPS-REMAIN-IN
   (PUSH BUDGET/
        (INCORP SEMLINKS (BUILDQ (COVERED/BY *)
                                 (GETP INTERP)))
        (ADDL VMODS (BUILDQ (PP (PREP IN)
                                *)))
        (TO S/POP)))


(S/WAS
   (PUSH T-DET-SG/
        (SPREAD (SEMQUANT SENSAR SEMHEAD SEMLINKS SENTEST SENFORM)
                (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/WAS-TRIP)))
```



```
(S/WAS-TRIP
   (WRD (LONG SHORT EXPENSIVE INEXPENSIVE NORMAL AVERAGE RECENT TYPICAL
              UNANTICIPATED UNBUDGETTED BUDGETTED EXPECT-ED UNEXPECTED
              ANTICIPATE-ED)
        (SCOPE (S/)
               (COND ((EQ (GETP TYPE)
                          (QUOTE DCL))
                      (SETR ACTION (BUILDQ (ADD: *** MODALITY *))))
                     (T (SETR ACTION
                              (BUILDQ (TEST: (GET: *** MODALITY *))))))))
        (ADDL VMODS (BUILDQ (PRED-ADJ *)))
        (TO S/POP)))


(S/WHAT
   (WRD (ARE WERE)
        (SETRQ V BE)
        (SETRQ VOICE ACTIVE)
        (SETRQ HEAD BE)
        (SETR TNS (CATP LEX V TNS))
        (SETR PRCODE (CATP LEX V PRCODE))
        (TO S/WHAT-ARE))
   (WRD (IS WAS)
        (SETRQ V BE)
        (SETRQ VOICE ACTIVE)
        (SETRQ HEAD BE)
        (SETR TNS (CATP LEX V TNS))
        (SETR PRCODE (CATP LEX V PRCODE))
        (TO S/WHAT-IS))
   (WRD (DO DID)
        (SETR TNS (CATP LEX V TNS))
        (TO S/WHAT-DO))
   (WRD HAVE
        (SETRQ TNS PAST)
        (SETRQ ASPECT PERFECT)
        (TO S/WHAT-HAVE)))


(S/WHAT-ARE
   (PUSH EX-PL-NOM-DET/
        (SETR SUBJ *)
        (SCOPE (S/WHAT)
               (VERIFY (PRCHECK (GETR SUBJ)
                                (GETR PRCODE))))
        (TO S/POP)))


(S/WHAT-DO
   (WRD (WE YOU)
        (SETR SUBJ (BUILDQ (PP (PRO *)
                               (FEATS (NU *)))
                           (CATP LEX PRO NUMBER)))
        (TO S/WHAT-DO-WE)))
```

```
: <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:66

(/.HAT-DO-WE
   (WRD HAVE
        (TO S/WHAT-HAVE-WE)))


(S/WHAT-HAVE
   (WRD WE
        (SETRQ SEMQUART THE)
        (SETRQ SEMHEAD DB/BUDGET)
        (ADDL SEMLINKS (QUOTE (BUDGET/OF SPEAKER)))
        (SETRQ ACTION (OUTPUT: (GET: *** MONEY/SPENT)))
        (SETRQ SDBJ (NP (PRO WE)
                        (FEATS (NU PL))))
        (TO S/WHAT-HAVE-WE)))


(S/WHAT-HAVE-WE
   (WRD (ALREADY RECENTLY)
        (ADDL VMODS (BDILDQ (ADV *)))
        (TO S/WHAT-WE-ADV))
   (JOMP S/WHAT-WE-HAVE ))


(S/WHAT-IS
   (JUMP S/WHAT-HAVE-WE
        (SCOPE (S/)
               (COND ((NULLR SUBJ)
                      (SETR SUBJ (BUILDQ (NP (PRC SOMEONE)
                                             (FEATS (NU SG)))))))))
   (PUSH EX-DET-SG-DEF/
        (SPREADR (SEMQUANT SENVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (SCOPE (S/WHAT)
               (VERIFY (PMCHECK (GETR SDBJ)
                                (GETR PMCODE))))
        (TO S/POP))
   (WRD IN
        (SCOPE (S/HOWMUCH)
               (SETR SDBJ (GETR OBJ))
               (SETR OBJ NIL))
        (TO S/HOWMUCH-IS-IN))
   (WRD LEFT
        (SETRQ V BE)
        (SCOPE (S/)
               (SETRQ ACTION (OUTPUT: (GET: *** MONEY/REMAINING))))
        (ADDL PREMODS (BUILDQ (ADJ LEFT)))
        (SETR LEFT *)
        (SETRQ VOICE ACTIVE)
        (SCOPE (S/WHAT-DO S/HOWMUCH)
               (COND ((GETR SDBJ)
                      (ADDL VMODS (BDILDQ (PP (PREP FOR)
                                              *)
                                          SDBJ))))
               (SETR SUBJ NIL))
        (TO S/WHAT-IS-LEFT)))
```

```
: <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:67

(S/WHAT-IS-BUDGET-FOR
   (PDSH MEETING/
        (ADDL WMODS (BUILDQ (PP (PREP FOR)
                                *)))
        (SCOPE (S/WHAT-DO S/WHAT-HAVE)
               (SETR SDBJ (NPBUILD)))
        (TO S/POP))
   (PDSH T-DET-SG-DET/
        (ADDL WMODS (BUILDQ (PP (PREP FOR)
                                *)))
        (SCOPE (S/WHAT-IS S/WHAT-ARE)
               (SETR SDBJ (NPBUILD)))
        (TO S/POP)))


(S/WHAT-IS-BUDGETED
   (OLD FOR
        (TO S/WHAT-IS-BUDGET-FOR)))


(S/WHAT-IS-IN
   (WRD THE
        (SETRQ SEMQUANT THE)
        (SETRQ DET (DET (ART THE)))
        (TO S/WHAT-IS-IN-THE)))


(S/WHAT-IS-IN-BUDGET
   (JUMP S/WHAT-IS-BUDGETED )
   (JUMP S/POP
        (SCOPE (S/WHAT-DO S/WHAT-HAVE)
               (VERIFY (GETR LEFT))
               (SETR SDBJ (NPBUILD)))))


(S/WHAT-IS-IN-THE
   (WRD BUDGET
        (SETRQ SEMHEAD DB/BUDGET)
        (PBDRY)
        (SETR W *)
        (SETR HEAD (BUILDQ (W *)))
        (SETRQ NU SG)
        (TO S/WHAT-IS-IN-BUDGET)))
```

```
(S/WHAT-IS-LEFT
   (WRD IN
        (TO S/WHAT-IS-IN))
   (JUMP S/POP
        (SETRQ SEMHEAD DB/BUDGET)
        (SCOPE (S/WHAT-HAVE-BE)
               (VERIFY (GETR LEFT)))
        (SETRQ DET (DET (ART THE)))
        (SETRQ HEAD (N BUDGET))
        (SETRQ NU SG)
        (SCOPE (S/WHAT-DO S/WHAT-HAVE S/WHAT-IS)
               (SETR SUBJ (NPBUILD)))))


(S/WHAT-BE-ADV
   (WRD SPENT
        (SETRQ VOICE PASSIVE)
        (SETR V (ROOTP * V))
        (TO S/POP)))


(S/WHAT-BE-HAVE
   (WRD BUDGET-ED
        (SETR HEAD (BUILDQ (N BUDGET)))
        (SETRQ DET (DET (ART THE)))
        (SETRQ NU SG)
        (SETRQ V BE)
        (SCOPE (S/)
               (COND ((NULLR TNS)
                      (SETRQ TNS PRESENT))))
        (SETRQ VOICE ACTIVE)
        (SCOPE (S/WHAT-DO S/WHAT-HAVE)
               (COND ((GETR SUBJ)
                      (ADDL NMODS (BUILDQ (PP (PREP FOR)
                                              *)
                                          SUBJ))
                      (SETR SUBJ NIL))))
        (TO S/WHAT-IS-BUDGETED))
   (WRD (LEFT REMAIN-ING)
        (ADLL PREMODS (BUILDQ (ADJ LEFT)))
        (SCOPE (S/WHAT)
               (VERIFY (NOT (EQ (GETR TNS)
                                (QUOTE PAST)))))
        (SCOPE (S/WHAT-DO S/WHAT-HAVE)
               (SETRQ V BE)
               (SETR LEFT *)
               (SETRQ VOICE ACTIVE)
               (ADDL NMODS (BUILDQ (PP (PREP FOR)
                                       *)
                                   SUBJ))
               (SETR SUBJ NIL))
        (TO S/WHAT-IS-LEFT))
```



```
   (JUMP S/WHAT-IS-LEFT
        (SCOPE (S/WHAT)
               (VERIFY (NOT (GETR TNS)
                            (QUOTE PAST))))
        (SCOPE (S/WHAT-DO S/WHAT-HAVE)
               (SETRQ V BE)
               (SETRQ VOICE ACTIVE)
               (ADDL NMODS (BUILDQ (PP (PREP FOR)
                                       *)
                                   SUBJ))
               (SETR SUBJ NIL))))


(S/WHEN
   (WRD (DID DOES)
        (SETR TNS (CATP LEX MODAL TNS))
        (SETRQ VOICE ACTIVE)
        (INCORP SEMLINKS (TIMEOF (GETR TNS)))
        (TO S/WHEN-DID)))


(S/WHEN-DID
   (PUSH MEETING/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/WHEN-DID-MEETING))
   (PUSH PEOPLE/
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (GETP INTERP)))
        (SETR SUBJ *)
        (TO S/WHEN-DID-PERSON))
   (CAT PRO-SUBJ
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (SEMPRO LEX)))
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NU *)))
                           (CATP LEX PRO NUMBER)))
        (TO S/WHEN-DID-PERSON)))


(S/WHEN-DID-MEETING
   (WRD (BEGIN START END)
        (COND ((EQ LEX (QUOTE END))
               (SETRQ ACTION (OUTPUT: (GET: *** END/TIME))))
              (T (SETRQ ACTION (OUTPUT: (GET: *** START/TIME)))))
        (SETR V (ROOTP LEX V))
        (SETR HEAD (GETR V))
        (TO S/POP)))
```

```
(S/WHEN-DID-PERSON
   (WRD (GO PLY TRAVEL)
        (AND (EQ LEX (QUOTE PLY))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR V *)
        (SETR HEAD *)
        (TO S/PERSON-GO))
   (WRD ATTEND
        (SETR V *)
        (SETR HEAD *)
        (TO S/ATTEND))
   (WRD VISIT
        (SETR V *)
        (SETR HEAD *)
        (TO S/PERSON-VISIT)))


(S/WHICH-TRIP
   (WRD (DID WILL)
        (SETRQ VOICE ACTIVE)
        (SETR TNS (CAPP LEX MODAL TNS))
        (SETR SENTNS (TIMEOP (GETR TNS)))
        (TO S/HOWMANY-TRIPS-WILL))
   (WRD HAS
        (SETRQ VOICE ACTIVE)
        (SETRQ TNS PAST)
        (TO S/HOWMANY-TRIPS-HAS)))


(S/WHO
   (WRD ATTEND-ED
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (SETR HEAD (ROOTP LEX V))
        (SETR V (GETR HEAD))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/ATTEND))
   (WRD (WERT TRAVEL-ED PLEW)
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (AND (EQ LEX (QUOTE PLEW))
             (INCORP SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR HEAD (ROOTP LEX V))
        (SETR V (GETR HEAD))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-GO))
   (WRD IS
        (SETRQ TNS PRESENT)
        (TO S/IS-PERSON))
```



```
   (WRD VISIT-ED
        (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW))))
        (SETR HEAD (ROOTP LEX V))
        (SETR V (GETR HEAD))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/PERSON-VISIT))
   (WRD WILL
        (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
        (SETRQ TNS FUTURE)
        (TO S/PERSON-WILL)))


(START/
   (WRD (WHEN WHERE)
        (SETR SEMTYPE *)
        (AND (EQ LEX (QUOTE WHERE))
             (SETR WHERE T))
        (SETRQ TYPE Q)
        (SETR QADV *)
        (TO S/QADV))
   (WRD (WREN WHY)
        (SETR SEMTYPE *)
        (SETRQ TYPE Q)
        (SETR QADV *)
        (TO S/WHEN))
   (WRD RHERE
        (SETR SEMTYPE *)
        (SETRQ TYPE Q)
        (SETR QADV *)
        (SETR WHERE T)
        (TO S/WHEN))
   (PUSH T-DETQ-PL/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SETR SUBJ *)
        (SETRQ TYPE Q)
        (TO S/HOWMANY-TRIPS))
   (PUSH T-DETQ-SG/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SETRQ TYPE Q)
        (SETR SUBJ *)
        (TO S/WHICH-TRIP))
   (PUSH BUDGET/
        (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
                 (CDR (GETP INTERP)))
        (SETRQ TYPE DCL)
        (SETR SUBJ *)
        (TO S/THE-BUDGET))
```

```
(PUSH P-DETQ-PL/
    (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMTEST SEMFORM)
             (CDR (GETP INTERP)))
    (SETP SUBJ *)
    (SETRQ TYPE Q)
    (TO S/HOWMANY-PEOPLE))
(WRD WHO
    (SETRQ ACTION (OUTPUT: (GET: *** TRAVELER)))
    (SETRQ SEMTYPE WHO)
    (SETRQ TYPE Q)
    (SETR SUBJ (BUILDQ (NP (PRO WHO)
                           (FEATS (NU SG/PL)))))
    (TO S/WHO))
(WRD THE
    (SETRQ TYPE DCL)
    (SETR DET (BUILDQ (DET (ART *))))
    (TO S/THE))
(WRD WHAT
    (SETR OBJ (BUILDQ (NP (PRO WHAT)
                          (FEATS (NU SG/PL)))))
    (SETRQ TYPE Q)
    (TO S/WHAT))
(PUSH T-DET-SG/
    (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
             (CDR (GETP INTERP)))
    (SETRQ TYPE DCL)
    (SETR SUBJ *)
    (TO S/DCL-SUBJ-SG))
(PUSH EX-DET-SG/
    (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
             (CDR (GETP INTERP)))
    (SETRQ TYPE DCL)
    (SETR SUBJ *)
    (TO S/DCL-SUBJ-SG))
(PUSH T-DET-PL/
    (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
             (CDR (GETP INTERP)))
    (SETRQ TYPE DCL)
    (SETR SUBJ *)
    (TO S/DCL-SUBJ-PL))
(PUSH EX-DET-PL/
    (SPREADR (SEMQUANT SEMVAR SEMHEAD SEMLINKS SEMFORM SEMTEST)
             (CDR (GETP INTERP)))
    (SETRQ TYPE DCL)
    (SETR SUBJ *)
    (TO S/DCL-SUBJ-PL))
(WRD I
    (SETR SUBJ (BUILDQ (NP (PRO I)
                           (FEATS (NU SG)))))
    (SETRQ TYPE DCL)
    (TO S/I))
```



```
(WRD AM
    (SETRQ SEMTYPE Y/N)
    (SETRQ TYPE Q)
    (TO S/AM))
(WRD IS
    (SETRQ SEMTYPE Y/N)
    (SETRQ TYPE Q)
    (SETRQ V BE)
    (SETRQ TNS PRESENT)
    (SETRQ VOICE ACTIVE)
    (TO S/IS))
(WRD DID
    (SETRQ SEMTYPE Y/N)
    (ADDL SEMLINKS (BUILDQ (TIME (BEFORE NOW))))
    (SETRQ TYPE Q)
    (SETRQ TNS PAST)
    (SETRQ VOICE ACTIVE)
    (TO S/DID))
(WRD DO
    (SETRQ SEMTYPE Y/N)
    (SETRQ TYPE Q)
    (SETRQ V DO)
    (SETRQ TNS PRESENT)
    (SETRQ VOICE ACTIVE)
    (TO S/DO))
(WRD (ARE WERE)
    (SETRQ SEMTYPE Y/N)
    (ADDL SEMLINKS (BUILDQ (TIME (BEFORE NOW))))
    (SETRQ TYPE Q)
    (SETRQ V BE)
    (SETR TNS (CATP LEX V TNS))
    (SETRQ VOICE ACTIVE)
    (TO S/ARE))
(WRD PLEASE
    (SETRQ TYPE IMP)
    (SETR SUBJ (BUILDQ (NP (PRO YOU)
                           (FEATS (NU SG)))))
    (SETRQ TNS PRESENT)
    (SETRQ VOICE ACTIVE)
    (TO S/IMP))
(JUMP S/IMP
    (SETRQ TYPE IMP)
    (SETR SUBJ (BUILDQ (NP (PRO YOU)
                           (FEATS (NU SG)))))
    (SETRQ TNS PRESENT)
    (SETRQ VOICE ACTIVE))
(WRD NOW
    (SETRQ TYPE Q)
    (TO S/NOW))
```

```
    (PUSH PEOPLE-PL/
        (ADDL SEMLINKS (BUILDQ (TRAVELER 0)
                               (GETP INTERP)))
        (SETRQ TYPE DCL)
        (SETR SUBJ *)
        (TO S/PEOPLE))
    (WRD (WE THEY)
        (ADDL SEMLINKS (BUILDQ (TRAVELER 0)
                               (SEMPRO LEX)))
        (SETRQ TYPE DCL)
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS PL))))
        (TO S/PEOPLE))
    (WRD (HE SHE I)
        (ADDL SEMLINKS (BUILDQ (TRAVELER 0)
                               (SEMPRO LEX)))
        (SETRQ TYPE DCL)
        (SETR SUBJ (BUILDQ (NP (PRO *)
                               (FEATS (NUMBER SG)))))
        (TO S/PERSON))
    (PUSH PERSON/
        (ADDL SEMLINKS (BUILDQ (TRAVELER 0)
                               (GETP INTERP)))
        (SETRQ TYPE DCL)
        (SETR SUBJ *)
        (TO S/PERSON))
    (WRD WAS
        (SETRQ SENTYPE Y/N)
        (SETRQ TYPE Q)
        (SETR V (ROOTP LEX V))
        (SETR HEAD (GETR V))
        (SETRQ TNS PAST)
        (SETRQ VOICE ACTIVE)
        (TO S/WAS))
    (WRD DOES
        (SETRQ SENTYPE Y/N)
        (SETRQ TYPE Q)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (TO S/DOES)))


(T-ALL-OF?/
    (WRD OF
        (TO T-PL-NON-DEF/)))


(T-DET-DEF-OR-SG/
    (JUMP T-DET-PL-DEF/ )
    (JUMP T-DET-SG/ ))
```



```
(T-DET-PL-DEF/
    (WRD ALL
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (QUANT *))
                          LEX))
        (TO T-PL/DET))
    (WRD ALL
        (SETR SUPERDET (BUILDQ (DET (QUANT *))
                               LEX))
        (TO T-PL-NON-DEF/))
    (WRD (ALL SOME ANY)
        (COND ((EQ LEX (QUOTE ALL))
               (SETR SEMQUANT *))
              (T (SETRQ SEMQUANT SOME)))
        (SETR SDPERDET (BUILDQ (DET (QUANT *))
                               LEX))
        (TO T-ALL-OF?/))
    (JUMP T-PL-NON-DEF/ ))


(T-DET-PL-INDEF/
    (WRD (SOME ANY)
        (SETRQ SEMQUANT SOME)
        (SETR DET (BUILDQ (DET (QUANT *))
                          LEX))
        (TO T-PL/DET))
    (JUMP T-PL/DET ))


(T-DET-PL/
    (JUMP T-DET-PL-DEF/ )
    (JUMP T-DET-PL-INDEF/ ))


(T-DET-SG-DEF/
    (WRD (THIS THAT)
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (FOR: SOME X / (FINDQ: DB/TRIP)
                                      ; T : (INPCCDS? X)))))
        (SETR DET (BUILDQ (DET *)
                          LEX))
        (TO T-SG/DET))
    (JUMP T-SG-NON-DEF/ )
    (WRD EACH
        (SETRQ SEMQUANT *)
        (SETR DET (BUILDQ (DET *)))
        (TO T-SG/DET))
    (WRD TRIP
        (PBODY)
        (TO TRIP/NUMBER*)))
```

```
(T-DET-SG-INDEF/
   (WRD (A AN)
        (SETRQ SEMQUANT 1)
        (SETRQ ARTAGREE T)
        (SETR DET (BUILDQ (DET (ART *))
                          LEX))
        (TO T-SG/DET))
   (WRD (ANY ANOTHER)
        (SETRQ SEMQUANT SOME)
        (SETR DET (BUILDQ (DET (QUANT *))
                          LEX))
        (TO T-SG/DET)))


(T-DET-SG/
   (JUMP T-DET-SG-DEF/ )
   (JUMP T-DET-SG-INDEF/ ))


(T-DETQ-PL/
   (WRD (WHAT WHICH)
        (SETRQ SEMQUANT EVERY)
        (SETR DET (BUILDQ (DET *)
                          LEX))
        (TO T-PL/DET))
   (WRD HOW
        (TO T-PL/MANY?)))


(T-DETQ-SG/
   (WRD (WHAT WHICH)
        (SETRQ SEMQUANT EVERY)
        (SETR DET (BUILDQ (DET *)
                          LEX))
        (TO T-SG/DET)))


(T-DO-YOU-HAVE/
   (JUMP T-DET-PL-INDEF/ )
   (JUMP T-PL-NON-DEM/ )
   (JUMP T-SG-NON-DEM/ )
   (JUMP T-DET-SG-INDEF/ ))
```



```
(T-ENTER/
   (WRD THAT
        (VERIFY (ASKFORK (QUOTE TRIP)
                         (QUOTE (FOR: SOME X / (FINDQ: DB/TRIP)
                                              ; T : (IMPCCUS? X)))))
        (SETR DET (BUILDQ (DET *)))
        (SETR SEMQUANT *)
        (TO T-SG/DET))
   (PUSH NUM/
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (QUANT (INTEGER *)))))
        (TO T-PL/NUM))
   (WRD (A AN)
        (SETRQ SEMQUANT 1)
        (SETR ARTAGREE T)
        (SETR DET (BUILDQ (DET (ART *))))
        (TO T-SG/DET)))


(T-PL-NON-DEM/
   (WRD (THE THESE THOSE)
        (COND ((EQ LEX (QUOTE THE))
               (SETRQ SEMQUANT ALL))
              (T (SETR SEMQUANT *)))
        (SETR DET (BUILDQ (DET *)
                          LEX))
        (TO T-PL/DET))
   (CAT POSS
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (SEMPRO LEX)))
        (SETR DET (BUILDQ (DET (POSS *))
                          LEX))
        (TO T-PL/DET))
   (PUSH NAME/
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (GETP INTERP)))
        (SETRQ DET (DET THE))
        (SETRQ PERSONHOD T)
        (SETR PERSON *)
        (TO T-PL-POSS?/)))


(T-PL-POSS?/
   (WRD -S
        (SCOPE (T-PL-NON-DEM/)
               (ADDL WHODS (BUILDQ (PP (PREP FOR)
                                       (NP *))
                                   PERSON)))
        (TO T-PL/DET)))
```

```
(T-PL/DET
   (PUSH NUM/
        (VERIFY (NEQ (GETR SEMQUANT)
                     (QUOTE SOME)))
        (SCOPE (T-DET-PL-INDEF/ T-DET-PL-DEF/)
               (COND ((NULLR SEMQUANT)
                      (SETR SEMQUANT (BUILDQ (THE *))))
                     (T (* ASK CHIP ABOUT THIS))))
        (SCOPE (T-DETQ-PL/ T-DET-PL-INDEF/ T-DET-PL-DEF/)
               (VERIFY (NEQ (GETR DET)
                            (QUOTE NOR@MANY))))
        (ADDL PREMODS (BUILDQ (ADJ *)))
        (TO T-PL/NUM))
   (JUMP T-PL/NUM ))


(T-PL/MANY?
   (RRD MANY
        (SETRQ SEMQUANT NOR@MANY)
        (SETRQ DET (DET (QUANT NOR@MANY)))
        (TO T-PL/NUM)))


(T-PL/NUM
   (CAT TRIP-ADJ
        (SELECTQ LEX ((UPCOMING UNTAKEN REMAINING OUTSTANDING FUTURE)
                      (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW))))
                      (SETRQ TRIPTIME FUTURE))
                     ((TAKEN RECENT PAST)
                      (SETRQ TRIPTIME PAST)
                      (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW)))))
                     (NEXT (SETRQ SEMQUANT (ORDINAL NEXT))
                           (SETRQ TRIPTIME FUTURE))
                     ((INTERNATIONAL FOREIGN DOMESTIC EUROPEAN)
                      (SETR SEMTEST (BUILDQ (GET: *** DESTINATION))))
                     (INCORP SEMLINKS (BUILDQ (MODALITY *))))
        (SCOPE (T-DETQ-PL/ T-DET-PL-INDEF/ T-DET-PL-DEF/ T-PL-NOR-DEF/
                           T-ENTER/)
               (VERIFY (DETADJCHECK (GETR DET)
                                    *
                                    (GETR TADJS)))
               (ADDL PREMODS (BUILDQ (@ (ADJ)
                                        # #)
                                     FEATURES
                                     (LIST *)))
               (ADDL TADJS *))
        (TO T-PL/NUM))
```



```
   (WRD TRIP-S
        (SETRQ SEMHEAD DB/TRIP)
        (SCOPE (T-DETQ-PL/ T-PL/DET T-ENTER/)
               (VERIFY (TRIP-ADJ-CK (GETR TADJS) ))
               (* SET TENSE WITH FUNCTION TRIP-ADJ-TNS))
        (SETR HEAD (BUILDQ (# TRIP)))
        (SETRQ NU PL)
        (SCOPE (T-DETQ-PL/ T-DET-PL-INDEF/ T-DET-PL-DEF/ T-DET-PL/
                           T-DO-YOU-HAVE/ T-ENTER/)
               (COND ((NULLR DET)
                      (SETRQ DET (DET)))
               (COND ((NULLR SEMQUANT)
                      (SETRQ SEMQUANT EVERY))))
        (TO TRIP/HEAD)))


(T-SG-NON-DEF/
   (WRD THE
        (SETR SEMQUANT *)
        (SETR DET (BUILDQ (DET (ART THE))))
        (TO T-SG/DET))
   (CAT POSS
        (SETRQ SEMQUANT THE)
        (ADDL SEMLINKS (BUILDQ (TRAVELER #)
                               (SEMPRO LEX)))
        (SETR DET (BUILDQ (DET (POSS #))
                          LEX))
        (TO T-SG/DET))
   (PUSH NAME/
        (SETRQ SEMQUANT THE)
        (ADDL SEMLINKS (BUILDQ (TRAVELER #)
                               (GETP INTERP)))
        (SETRQ DET (DET THE))
        (SETR PERSONMOD T)
        (SETR PERSON *)
        (TO T-SG-POSS?/)))


(T-SG-POSS?/
   (WRD -S
        (SCOPE (T-SG-NOR-DEF/)
               (SETRQ DET (DET (ART THE)))
               (ADDL MODS (BUILDQ (PP (PREP FOR)
                                      (NP *))
                                  PERSON)))
        (TO T-SG/DET)))
```

```
(T-SG/DET
   (CAT TRIP-ADJ
      (SELECTQ LEX ((UPCOMING UNTAKEN REMAINING OUTSTANDING FUTURE)
                    (SETRQ TRIPTIME FUTURE)
                    (INCORP SEMLINKS (QUOTE (TIME (AFTER NOW)))))
                   ((TAKEN RECENT PAST)
                    (SETRQ TRIPTIME PAST)
                    (INCORP SEMLINKS (QUOTE (TIME (BEFORE NOW)))))
                   (NEXT (SETRQ TRIPTIME FUTURE)
                         (SETRQ SEMQUANT (ORDINAL NEXT)))
                   ((INTERNATIONAL FOREIGN DOMESTIC EUROPEAN)
                    (SETR SEMTEST (BUILDQ (GET: *** DESTINATION))))
                   (INCORP SEMLINKS (BUILDQ (MODALITY *)))))
      (SCOPE (T-ENTER/ T-DET-SG-INDEF/ T-DET-SG-DEF/ T-DETQ-SG/
                       T-SG-NOW-DEM/ T-SG-POSS?/)
             (COND ((GETR ARTAGREE)
                    (VERIFY (ARTAGREE (GETR DET)
                                      LEX))
                    (SETR ARTAGREE NIL)))
             (ADDL TADJS *))
      (ADDL PREMODS (BUILDQ (@ (ADJ)
                               @ @)
                            FEATURES
                            (LIST *)))
      (TO T-SG/DET))
   (WRD TRIP
      (SETRQ SEMHEAD DB/TRIP)
      (SCOPE (T-ENTER/ T-DET-SG-INDEF/ T-DET-SG-DEF/ T-DETQ-SG/
                       T-SG-POSS?/ T-SG-NOW-DEF/)
             (COND ((GETR ARTAGREE)
                    (VERIFY (TRIP-ADJ-CK (GETR TADJS)))
                    (* SET TENSE FOR THINGS LIKE "UPCOMING TRIP")
                    (VERIFY (ARTAGREE (GETR DET)
                                      LEX))
                    (SETR ARTAGREE NIL))))
      (SETR HEAD (BUILDQ (N TRIP)))
      (SETRQ NU SG)
      (SCOPE (T-DET-SG-INDEF/ T-DET-SG-DEF/ T-ENTER/ T-SG-NOW-DEM/
                              T-DETQ-SG/)
             (COND ((NULLR DET)
                    (SETRQ DET (DET)))
             (COND ((NULLR SEMQUANT)
                    (SETRQ SEMQUANT EACH))))
      (TO TRIP/HEAD)))


(THE-PROJECT/
   (WRD THE
      (SETR DET (BUILDQ (DET (ART *))))
      (TO THE-PROJECT/THE)))
```



```
(THE-PROJECT/P
   (WRD PROJECT-N
      (SETR HEAD (BUILDQ (N *)))
      (SETRQ NU SG)
      (TO THE-PROJECT/POP)))


(THE-PROJECT/POP
   (POP (NPBUILD)
      (SCOPE (THE-PROJECT/)
             (LIFTR INTERP))))


(THE-PROJECT/THE
   (PUSH PROJECT/
      (ADDL PREMODS *)
      (TO THE-PROJECT/P)))


(TO-PLACE/
   (WRD TO
      (TO TO-PLACE/TO)))


(TO-PLACE/TO
   (PUSH MEETING/
      (SETR SEMLINKS (BUILDQ (TO/ATTEND @)
                             (GETF INTERP)))
      (SETR PP (BUILDQ (PP (PREP TO)
                           *)))
      (TO PP/POP))
   (PUSH CITY/
      (SETR SEMLINKS (BUILDQ (DESTINATION @)
                             (GETF INTERP)))
      (SETR PP (BUILDQ (PP (PREP TO)
                           (NP @))))
      (TO PP/POP)))
```

```
(TO/
   (WRD ATTEND
        (PBODY)
        (SETR V *)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (TO TO/ATTEND))
   (WRD TAKE
        (PBODY)
        (SETR V *)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (SETR SUBJ (BUILDQ (NP (PRO SOMEONE)
                               (FEATS (NU SG)))))
        (TO TO/TAKE))
   (WRD (GO FLY TRAVEL)
        (PBODY)
        (AND (EQ LEX (QUOTE FLY))
             (ADDL SEMLINKS (QUOTE (MODE/OF/TRANSPORT AIR))))
        (SETR V *)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (SETR SUBJ (BUILDQ (NP (PRO SOMEONE)
                               (FEATS (NU SG)))))
        (TO TO/GO))
   (WRD SEND
        (PBODY)
        (SETR V *)
        (SETR HEAD *)
        (SETRQ TNS PRESENT)
        (SETRQ VOICE ACTIVE)
        (SETR SUBJ (BUILDQ (NP (PRO SOMEONE)
                               (FEATS (NU SG)))))
        (TO TO/SEND)))


(TO/ATTEND
   (PUSH MEETING/
        (SETR OBJ *)
        (TO TO/POP)))


(TO/GO
   (WRD THERE
        (ADDL SEMLINKS (BUILDQ (DESTINATION *)))
        (ADDL PMODS (BUILDQ (PP (PREP TO)
                                (NP (PRO THERE)
                                    (FEATS (NU SG))))))
        (SETR THERE T)
        (TO TO/GO-PLACE))
   (JUMP TO/GO-PLACE ))
```



```
(TO/GO-PLACE
   (JUMP TO/POP )
   (PUSH TO-PLACE/
        (AND (GETP INTERP)
             (MERGE SEMLINKS (GETP INTERP)))
        (AND (GETP SEMLINKS)
             (ADDL SEMLINKS (GETP SEMLINKS)))
        (SCOPE (TO/GO)
               (VERIFY (NULLR THERE))
               (ADDL PMODS *))
        (TO TO/GO-TO)))


(TO/GO-TO
   (JUMP TO/GO-WITH-TO ))


(TO/GO-WITH-TO
   (PUSH DATMOD-EXTENT/
        (ADDL SEMLINKS (BUILDQ (TIME *)))
        (SCOPE (TO/)
               (VERIFY (TIMESENSE (GETR TNS)
                                  *))
               (ADDL PMODS *))
        (TO TO/POP))
   (JUMP TO/POP ))


(TO/POP
   (POP (SBUILD)
        (SCOPE (TO/)
               (LISTR SEMLINKS))))


(TO/SEND
   (PUSH PERSON/
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (GETP INTERP)))
        (SETR OBJ *)
        (TO TO/GO))
   (CAT PRO-OBJ
        (ADDL SEMLINKS (BUILDQ (TRAVELER *)
                               (SEMPRO LEX)))
        (SETR OBJ (BUILDQ (NP (PRO *)
                              (FEATS (NU *)))
                          (CATP LEX PRO NUMBER)))
        (TO TO/GO))
   (CAT DIGITS
        (PBODY)
        (ADDL SEMLINKS (BUILDQ (TRAVELER (! * # PERSON))
                               (GENSYM)))
        (VERIFY (NOT (EQ * 1)))
        (SETR PREMODS (BUILDQ (POSTART #)
                              *))
        (TO TO/SEND-NUM)))
```

```
(TO/SEND-NUM
   (WRD PEOPLE
        (SETR OBJ (BUILDQ (NP * (N *)
                                (FEATS (NU PL)))
                          PRENODS
                          (ROOTP * N)))
        (TO TO/GO)))


(TO/TAKE
   (WRD A
        (TO TO/TAKE-A)))


(TO/TAKE-A
   (WRD TRIP
        (PBDRY)
        (SETR OBJ (BUILDQ (NP (DET (ART A))
                              (N TRIP)
                              (FEATS (NU SG)))))
        (TO TO/GO)))


(TO/VISIT
   (PUSH CITY/
        (ADDL SEMLINKS (BUILDQ (DESTINATION *)
                               (GETP INTERP)))
        (SETR OBJ *)
        (TO TO/GO-TO)))


(TRIP-BY/
   (WRD BY
        (TO TRIP/BY)))


(TRIP-DET/
   (JUMP T-DET-PL/ )
   (JUMP T-DET-SG/ ))


(TRIP-FOR/
   (WRD FOR
        (TO TRIP/FOR)))


(TRIP-TO/
   (WRD TO
        (TO TRIP/TO)))
```



```
(TRIP-TO/POP
   (POP (GETR PP)
        (SCOPE (TRIP-TO/ TRIP-BY/ TRIP-FOR/)
               (LIFTR INTERP)
               (LIFTR SEMLINKS))))


(TRIP/BY
   (CAT CONVEYANCE
        (PBDRY)
        (SETR INTERP (BUILDQ (MODE/OF/TRANSPORT *)))
        (SETR PP (BUILDQ (PP (PREP BY)
                             (NP (N *)
                                 (FEATS (NU SG))))))
        (TO TRIP-TO/POP)))


(TRIP/BY-MOD
   (PUSH TRIP-TO/
        (ADDR SEMLINKS (GETP INTERP))
        (ADDL WHODS *)
        (TO TRIP/BY-TO-MOD))
   (JUMP TRIP/BY-TO-MOD ))


(TRIP/BY-TO-MOD
   (PUSH TRIP-FOR/
        (ADDR SEMLINKS (GETP INTERP))
        (ADDL WHODS *)
        (TO TRIP/DATE?))
   (JUMP TRIP/DATE? ))


(TRIP/DATE
   (JUMP TRIP/POP )
   (PUSH TRIP-BY/
        (ADDL SEMLINKS (GETP INTERP))
        (ADDL WHODS *)
        (TO TRIP/POP)))


(TRIP/DATE?
   (PUSH DATEMOD/
        (ADDL SEMLINKS (GETP INTERP))
        (ADDL WHODS *)
        (TO TRIP/POP))
   (JUMP TRIP/POP ))
```

```
(TRIP/FOR
   (PUSH PEOPLE/
       (INCORP SEMLINKS (BUILDQ (TRAVELER 0)
                                (GETP INTERP)))
       (SETR PP (BUILDQ (PP (PREP FOR)
                            *)))
       (TO TRIP-TO/POP))
   (CAT PRO-OBJ
       (INCORP SEMLINKS (BUILDQ (TRAVELER 0)
                                (SEMPRO LEX)))
       (SETR PP (BUILDQ (PP (PREP FOR)
                            (NP (PRO *)
                                (FEATS (NU 0))))
                        (CATP LEX PRO NUMBER)))
       (TO TRIP-TO/POP))
   (PUSH MEETING/
       (SETR INTERP (BUILDQ (TO/ATTEND 0)
                            (GETP INTERP)))
       (SETR PP (BUILDQ (PP (PREP TO)
                            *)))
       (TO TRIP-TO/POP)))


(TRIP/FOR-MOD
   (JUMP TRIP/DATE? )
   (PUSH TRIP-BY/
       (ADDL SEMLINKS (GETP INTERP))
       (ADDL MMODS *)
       (TO TRIP/DATE?))
   (PUSH TRIP-TO/
       (ADDL SEMLINKS (GETP INTERP))
       (ADDL MMODS *)
       (TO TRIP/DATE?)))


(TRIP/FOR-ONE
   (WRD PERSON
       (ADDL MMODS (BUILDQ (PP (PREP FOR)
                               (NP (DET (QUANT (INTEGER 1)))
                                   (N PERSON)
                                   (FEATS (NU SG))))))
       (TO TRIP/HEAD)))
```



```
(TRIP/HEAD
   (PUSH TRIP-FOR/
       (COND ((GETP INTERP)
              (INCORP SEMLINKS (BUILDQ (TRAVELER 0)
                                       (GETP INTERP))))
             (T (MERGE SEMLINKS (GETP SEMLINKS) )))
       (ADDL MMODS *)
       (TO TRIP/FOR-MOD))
   (PUSH TRIP-TO/
       (ADDL SEMLINKS (GETP INTERP))
       (ADDL MMODS *)
       (TO TRIP/TO-MOD))
   (PUSH TRIP-BY/
       (ADDL SEMLINKS (GETP INTERP))
       (ADDL MMODS *)
       (TO TRIP/BY-MOD))
   (PUSH DATEMOD/
       (ADDL SEMLINKS (GETP INTERP))
       (ADDL MMODS *)
       (TO TRIP/DATE))
   (WRD -PAUSE-
       (TO TRIP/POP))
   (JUMP TRIP/POP ))


(TRIP/NUMBER1
   (WRD NUMBER
       (TO TRIP/NUMBER2))
   (JUMP TRIP/NUMBER2 ))


(TRIP/NUMBER2
   (PUSH TRIPNUM/
       (ADDL SEMLINKS (BUILDQ (TRIP# *)))
       (SETR TRIPNUM *)
       (TO TRIP/TRIPNUMBER)))


(TRIP/PEOPLENUM
   (WRD PEOPLE
       (SCOPE (T-DET-PL-DEF/ T-DET-SG-INDEF/ T-DET-SG-DEF/ T-ENTER/
                             T-SG-MOD-DEM/ T-DET-PL-INDEF/
                             T-DO-YOU-HAVE/ T-DETQ-PL/ T-DETQ-SG/)
              (ADDL MMODS
                    (BUILDQ (PP (PREP FOR)
                                (NP (DET (QUANT (INTEGER *)))
                                    (N PERSON)
                                    (FEATS (NU PL))))
                            PEOPLENUM)))
       (TO TRIP/HEAD)))
```

```
(TRIP/POP
   (POP (COND ((NULLR SUPERDET)
               (NPBUILD))
              (T (BUILDQ (NP * (N (PNO ONE))
                                 (PP (PREP OF)
                                     *)
                                 (PEATS (NO PL)))
                         SUPERDET
                         (NPBUILD))))
        (SCOPE T (SETR SEMLINKS (CLEAR. M (GETR SEMLINKS)))
               (SETR INTERP (BUILDQ (! # # DB/TRIP *)
                                    (OR (GETR SEQUANT)
                                        (QUOTE ALL))
                                    (GENSYM)
                                    SEMLINKS))
               (LIFTR INTERP))
        (SCOPE (T-PL/MOD T-SG/DET)
               (VERIFY (NOT (AND (GETR PERSONMOD)
                                 (GETR TRIPFOR)))))))


(TRIP/TO
   (PUSH CITY/    *
        (SETR INTERP (BUILDQ (DESTINATION #)
                             (GETP IN' ERP)))
        (SETR PP (BUILDQ (PP (PREP TO)
                             (NP *))))
        (TO TRIP-TO/POP))
   (PUSH MEETING/
        (SETR INTERP (BUILDQ (TO/ATTEND #)
                             (GETP INTERP)))
        (SETR PP (BUILDQ (PP (PREP TO)
                             *)))
        (TO TRIP-TO/POP))
   (CAT NPR
        (VERIFY (NOT (CATCHECK LEX (QUOTE SPONSOR))))
        (PUSH)
        (SETR INTERP (BUILDQ (DESTINATION *)))
        (SETR PP (BUILDQ (PP (PREP TO)
                             (NP (NPR (LOCATION *)))))))
        (TO TRIP-TO/POP)))


(TRIP/TO-FOR-MOD
   (PUSH TRIP-BY/
        (ADD SEMLINKS (GETP INTERP))
        (ADDL MODS *)
        (TO TRIP/DATE?))
   (JUMP TRIP/DATE? ))
```



```
(TRIP/TO-MOD
   (PUSH TRIP-FOR/
        (ADDL SEMLINKS (BUILDQ (TRAVELER #)
                               (GETP INTERP)))
        (ADDL MODS *)
        (TO TRIP/TO-FOR-MOD))
   (JUMP TRIP/TO-FOR-MOD ))


(TRIP/TRIPNUMBER
   (POP (GETR TRIPNUM)
        (SCOPE T (SETR INTERP (BUILDQ (! THE # DE/TRIP *)
                                      (GENSYM)
                                      SEMLINKS))
               (LIFTR SEMLINKS)
               (LIFTR INTERP))))


(TRIPNUM/
   (PUSH DIGIT-STRING/
        (VERIFY (EQ (GETP DIGITSTRINGPEATS)
                    (QUOTE TRIPNO)))
        (SETR TRIPNUM (NUMBOF *))
        (TO TRIPNUM/POP))
   (PUSH NUM/
        (VERIFY (IGREATERP (NUMBOF *)
                           9))
        (SETR TRIPNUM (ITIMES 100 (NUMBOF *)))
        (TO TRIPNUM/NUM)))


(TRIPNUM/NUM
   (PUSH NUM/
        (SCOPE (TRIPNUM/)
               (VERIFY (IGREATERP (NUMBOF *)
                                  9))
               (SETR TRIPNUM (IPLUS (GETR TRIPNUM)
                                    (NUMBOF *))))
        (TO TRIPNUM/POP))
   (END ON
        (TO TRIPNUM/ON)))


(TRIPNUM/ON
   (CAT DIGITS
        (VERIFY (NOT (PHRSE LEX (QUOTE (ON ZERO)))))
        (SCOPE (TRIPNUM/)
               (SETR TRIPNUM (IPLUS (GETR TRIPNUM)
                                    (NUMBOF *))))
        (TO TRIPNUM/POP)))
```

```
: <SPSYS>MIDGRAM.;21   Fri 29-Oct-76 1:33PM          PAGE 1:99

(TRIPNUM/POP
   (POP (GETR TRIPNUM) ))


(YEAR-DATE/
   (WRD MINETEEN
        (SETQ YEAR 1900)
        (TO YEAR-DATE/NUM))
   (JUMP YEAR-DATE/NUM
        (PBOPY)
        (SETQ YEAR 0)
        (SETQ YEARFLAG T)))


(YEAR-DATE/LAST
   (CAT DIGITS
        (SCOPE (YEAR-DATE/)
               (SETR YEAR (IPLUS (NUMBOF *)
                                 (GETR YEAR)
                                 (COND ((GETR YEARFLAG)
                                        1900)
                                       (T 0)))))
        (TO YEAR-DATE/POP))
   (JUMP YEAR-DATE/POP
        (SCOPE (YEAR-DATE/)
               (VERIFY (NULLR YEARFLAG)))))


(YEAR-DATE/NUM
   (WRD (SIXTY SEVENTY)
        (SCOPE (YEAR-DATE/)
               (SETR YEAR (IPLUS (NUMBOF *)
                                 (GETR YEAR))))
        (TO YEAR-DATE/LAST)))


(YEAR-DATE/POP
   (POP (GETR YEAR) ))
)
(RPAQQ GRAMMARNAME MIDGRAM)

(RPAQQ LOADUPNAME SYMPORK)
(DECLARE: DONTCOPY
  (FILEMAP (MIL)))
STOP
```

Appendix 2 - Sample Parses

The following sentences were parsed in text mode using BIGDICT and BIGGRAM. For each sentence the syntactic parse tree is shown followed by the semantic interpretation which is also created by the parser. The optimized interpretation is obtained from the original interpretation by a function in the TRIP fork (see Vol. 5, Sec. E.3 for details).

Comments which follow the output are enclosed in brackets, {}, and were added to explain or call attention to interesting aspects of the parsing process and its result. Additional examples can be found in Vol. 5, Sec. I.2.

```
**************************
*                        *
* DO WE HAVE A SURPLUS   *
*                        *
**************************
```

Found complete parse:

```
S Q
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V HAVE
     OBJ NP N SURPLUS
            FEATS NU SG
```

Interpretation:

```
[FOR: SOME A0027 / (FINDQ: DB/GROUP (MEMBERS SPEAKER))
      : T ; (FOR: SOME A0026 / (FINDQ: DB/CONTRACT (GROUP A0027))
                  : T ; (FOR: THE A0028 / (FINDQ: DB/BUDGET
                                                  (BUDGET/OF A0026))
                              : T ;
                              (TEST: (GREATERP (GET: A0028
                                                     MONEY/REMAINING)
                                               0]
```

Optimized interpretation:

```
[FOR: SOME A0027 / (FIND: (INSTANCE/OF (QUOTE DB/GROUP))
                          (MEMBERS (QUOTE SPEAKER)))
      : T ; (FOR: SOME A0026 / (FIND: (INSTANCE/OF (QUOTE DB/CONTRACT))
                                      (GROUP A0027))
                  : T ;
                  (FOR: THE A0028 / (FIND: (INSTANCE/OF (QUOTE DB/BUDGET))
                                           (BUDGET/OF A0026))
                        : T ;
                        (TEST: (GREATERP (GET: A0028
                                               (QUOTE MONEY/REMAINING))
                                         (QUOTE 0]
```

{This is an example of the type of expansion (of the word "we") which is usually left for the TRIP fork, but which can be done by the grammar. The interpretation could have been built:

```
(FOR THE A0027 / (FINDQ: DB/BUDGET (BUDGET/FOR SPEAKER))
      : T ; (TEST: (GREATERP (GET: A0027 MONEY(REMAINING)
                             0)))
```

if the TRIP fork could infer the meaning of a person at the end of a BUDGET/FOR link. This shorter form is created in other places in the grammar.}

```
**********************
*                    *
* SHOW ME HER TRIP-S *
*                    *
**********************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V SHOW
     OBJ NP DET POSS HER
            N TRIP
            FEATS NU PL
```

Interpretation:

```
(FOR: THAT A0021 / (FINDQ: DB/PERSON (GENDER FEMALE))
      : T ; (FOR: EVERY A0022 / (FINDQ: DB/TRIP (TRAVELER A0021))
                  : T ; (OUTPUT: A0022)))
```

Optimized interpretation:

```
(FOR: THAT A0021 / (FIND: (INSTANCE/OF (QUOTE DB/PERSON))
                          (GENDER (QUOTE FEMALE)))
      : T ; (FOR: EVERY A0022 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                                       (TRAVELER A0021))
                  : T ; (OUTPUT: A0022)))
```

{Third person pronouns are expanded into an interpretation with the quantifier THAT since it is necessary for the TRIP fork to have an antecedent if the interpretation is to execute correctly. Another way of handling this would be to call the TRIP fork as soon as the pronoun is found to verify the existence of an antecedent. This latter method is used for phrases like "that meeting".}

```
**********************************
*                                *
* WHICH TRIP-S WERE CANCEL-ED    *
*                                *
**********************************
```

```
Found complete parse:

     S Q
       SUBJ NP PRO SOMEONE
               FEATS NU SG
       AUX TNS PAST
           VOICE PASSIVE
       VP V CANCEL
          OBJ NP DET WHICH
                 N TRIP
                 FEATS NU PL


Interpretation:

     (FOR: EVERY A0013 / (FINDQ: DB/TRIP (MODALITY CANCELED)
                                 (TIME (BEFORE NOW)))
           : T ; (OUTPUT: A0013))


Optimized interpretation:

     (FOR: EVERY A0013 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                                (MODALITY (QUOTE CANCELED))
                                (TIME (QUOTE PAST)))
           : T ; (OUTPUT: A0013))
```

{"Which" causes the quantifier EVERY to be used, since it is assumed that at least one trip meets the criterion. If this is not the case in the data base, a message to that effect will be printed. If the quantifier ALL had been used in that circumstance, no action would be taken if no canceled trips were found.}

```
**************************
*                        *
* I WILL FLY TO SAN@DIEGO *
*                        *
**************************
```

```
Found complete parse:

     S DCL
       SUBJ NP PRO I
               FEATS NUMBER SG
       AUX TNS FUTURE
           VOICE ACTIVE
       VP " FLY
          PP PREP TO
             NP NPR LOCATION CITY SAN@DIEGO


Interpretation:

     (FOR: THE A0050 / (FINDQ: LOCATION (CITY SAN@DIEGO))
           : T ; (FOR: 1 A0051 / (BUILDQ: DB/TRIP (TIME (AFTER NOW))
                                          (DESTINATION A0050)
                                          (MODE/OF/TRANSPORT AIR)
                                          (TRAVELER SPEAKER))
                       : T ; T))


Optimized interpretation:

     (FOR: THE A0050 / (FINDLOC (INSTANCE/OF (QUOTE LOCATION))
                                (CITY (QUOTE SAN@DIEGO)))
           : T ; (FOR: 1 A0051 / (BUILD: DB/TRIP (TIME (QUOTE FUTURE))
                                         (DESTINATION A0050)
                                         (MODE/OF/TRANSPORT (QUOTE AIR))
                                         (TRAVELER (QUOTE SPEAKER)))
                       : T ; T))
```

{The mode of transport is recognized in adjectives ("the bus fare"), prepositional phrases ("by train"), and verbs ("fly"). First person pronouns always become SPEAKER in the interpretation for the convenience of the TRIP fork.}

```
**************************************
*                                    *
* SHOW ME BILL -S TRIP TO WASHINGTON *
*                                    *
**************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V SHOW
     OBJ NP DET ART THE
            N TRIP
            PP PREP FOR
               NP NPR NAME FIRST BILL
            PP PREP TO
               NP NPR LOCATION AMBIG WASHINGTON
            FEATS NU SG
```

Interpretation:

```
[FOR: THE A0069 / (FINDQ: PERSON (FIRSTNAME BILL))
      : T ; (FOR: THE A0070 / (FINDQ: LOCATION (AMBIG WASHINGTON))
                  : T ; (FOR: THE A0071 / (FINDQ: DB/TRIP
                                                  (DESTINATION A0070)
                                                  (TRAVELER A0069))
                              : T ; (OUTPUT: A0071]
```

Optimized interpretation:

```
[FOR: THE A0069 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (FIRSTNAME (QUOTE BILL)))
      : T ; (FOR: THE A0070 / (FINDLOC (INSTANCE/OF (QUOTE LOCATION))
                                       (AMBIG (QUOTE WASHINGTON)))
                  : T ; (FOR: THE A0071 / (FIND: (INSTANCE/OF
                                                   (QUOTE DB/TRIP))
                                                 (DESTINATION A0070)
                                                 (TRAVELER A0069))
                              : T ; (OUTPUT: A0071]
```

{"Washington" is ambiguous since it is both a state and a city. This fact is noted in both the parse tree and the interpretation. All verbs that request a printout ("print", "list", "give (me)," "show") affect the parse tree but not the interpretation.}

```
**************************************
*                                    *
* GIVE ME A LIST OF THE UNTAKEN TRIP-S *
*                                    *
**************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V GIVE
     OBJ NP DET THE
             ADJ UNTAKEN
             N TRIP
             FEATS NU PL
```

Interpretation:

```
(FOR: ALL A0081 / (FINDQ: DB/TRIP (TIME (AFTER NOW)))
      : T ; (OUTPUT: A0081))
```

Optimized interpretation:

```
(FOR: ALL A0081 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                         (TIME (QUOTE FUTURE)))
      : T ; (OUTPUT: A0081))
```

{The adjective "untaken" marks the trip as future. Other trip adjectives that would affect the tense are "recent", "taken", "upcoming", and "past". The grammar checks agreement so that phrases like "recent upcoming trip" and "She took an upcoming trip" are not accepted.}

```
**********************************************
*                                            *
* CANCEL LYN -S TRIP TO THE ASA MEETING      *
*                                            *
**********************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V CANCEL
     OBJ NP DET ART THE
            N TRIP
            PP PREP FOR
               NP NPR NAME FIRST LYN
            PP PREP TO
               NP DET ART THE
                  ADJ NP NPR ASA
                  N MEETING
                  FEATS NU SG
            FEATS NU SG
```

Interpretation:

```
[FOR: THE A0088 / (FINDQ: PERSON (FIRSTNAME LYN))
      : T ; (FOR: THE A0089 / (FINDQ: DB/CONFERENCE (SPONSOR ASA))
                  : T ; (FOR: THE A0090 / (FINDQ: DB/TRIP
                                                  (TO/ATTEND A0089)
                                                  (TRAVELER A0088))
                              : T ; (CANCEL; A0090]
```

Optimized interpretation:

```
[FOR: THE A0088 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (FIRSTNAME (QUOTE LYN)))
      : T ; (FOR: THE A0089 / (FIND: (INSTANCE/OF (QUOTE DB/CONFERENCE))
                                     (SPONSOR (QUOTE ASA)))
                  : T ; (FOR: THE A0090 / (FIND: (INSTANCE/OF
                                                   (QUOTE DB/TRIP))
                                                 (TO/ATTEND A0089)
                                                 (TRAVELER A0088))
                              : T ; (CANCEL; A0090]
```

{Here is one of the few occurrences of an action other than TEST: or OUTPUT:. The possessive is always interpreted as a traveler in this system.}

```
**********************************
*                                *
* HOW MANY TRIP-S HAS RICH TAKEN *
*                                *
**********************************
```

```
Found complete parse:

     S Q
       SUBJ NPR NAME FIRST RICH
       AUX INS PAST
           VOICE ACTIVE
       VP V TAKE
          CBJ NP DET QUANT HOW@MANY
                 N TRIP
                 FEATS NU PL
```

```
Interpretation:

     [FOR: THE A0093 / (FINDQ: PERSON (FIRSTNAME RICH))
           : T ; (FOR: THE A0092 / [SETOF: (FINDQ: DB/TRIP (TRAVELER A0093
                                                  (TIME (BEFORE NOW]
                       : T ; (OUTPUT: (COUNT: A0092]
```

```
Optimized interpretation:

     [FOR: THE A0093 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                              (FIRSTNAME (QUOTE RICH)))
           : T ; (FOR: THE A0092 / [SETOF: (FINDQ: DB/TRIP (TRAVELER A0093
                                                  (TIME (BEFORE NOW]
                       : T ; (OUTPUT: (COUNT: A0092]
```

{"How many" questions must result in interpretations which create a set and then count the elements. See text for further description of how the grammar handles this.}

```
**********************************************
*                                            *
* DOES THE SPEECH BUDGET HAVE A SURPLUS      *
*                                            *
**********************************************
```

Found complete parse:

```
S 0
  SUBJ NP DET ART THE
          SPEECH
          N BUDGET
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V HAVE
     OBJ NP N SURPLUS
            FEATS NU SG
```

Interpretation:

```
(FOR: THE A0104 / (FINDQ: DB/BUDGET (PROJECT (SPEECH)))
      : T ; (TEST: (GREATERP (GET: A0104 MONEY/REMAINING)
                             0)))
```

Optimized interpretation:

```
[FOR: THE A0104 / [FIND: (INSTANCE/OF 'QUOTE DB/BUDGET))
                         (PROJECT (QUOTE (SPEECH]
      : T ; (TEST: (GREATERP (GET: A0104 (QUOTE MONEY/REMAINING))
                             (QUOTE 0]
```

{The concept of a surplus is not directly available in the data base, so it must be calculated explicitly.}

```
**********************************************
*                                            *
* IS JOHN MAKHOUL SCHEDULE-ED TO GO TO PARIS *
*                                            *
**********************************************
```

Found complete parse:

```
S Q
  SUBJ NP PRO SOMEONE
          FEATS NU SG
  AUX TNS PRESENT
      VOICE PASSIVE
  VP V SCHEDULE
     OBJ S NIL
           SUBJ NPR NAME FIRST JOHN
                         LAST MAKHOUL
           AUX TNS PRESENT
               VOICE ACTIVE
           VP V GO
              PP PREP TO
                 NP NPR LOCATION CITY PARIS
```

Interpretation:

```
[TEST: (FOR: THE A0109 / (FINDQ: PERSON (LASTNAME MAKHOUL)
                                 (FIRSTNAME JOHN))
             : T ; (FOR: THE A0110 / (FINDQ: LOCATION (CITY PARIS))
                          : T ; (FOR: ALL A0111 /
                                      (FINDQ: DB/TRIP (DESTINATION A0110)
                                              (TRAVELER A0109))
                                      : T ; T]
```

Optimized interpretation:

```
[TEST: (FOR: THE A0109 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                                (LASTNAME (QUOTE MAKHOUL))
                                (FIRSTNAME (QUOTE JOHN)))
             : T ; (FOR: THE A0110 / (FINDLOC (INSTANCE/OF (QUOTE
                                                            LOCATION))
                                              (CITY (QUOTE PARIS)))
                          : T ; (FOR: ALL A0111 /
                                      (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                                             (DESTINATION A0110)
                                             (TRAVELER A0109))
                                      : T ; T]
```

{Yes/no questions may be answered by TEST: either as the primary form or as an action. The use of the passive voice complicates the parse tree but not the interpretation.}

```
**********************************************
*                                            *
* SCHEDULE A TRIP FOR JACK KLOVSTAD TO SDC *
*                                            *
**********************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V SCHEDULE
     OBJ NP DET ART A
            N TRIP
            PP PREP FOR
               NPR NAME FIRST JACK
                        LAST KLOVSTAD
            PP PREP TO
               NP NPR LOCATION SDC
            FEATS NU SG
```

Interpretation:

```
(FOR: THE A0114 / (FINDQ: PERSON (LASTNAME KLOVSTAD)
                                 (FIRSTNAME JACK))
      : T ; (FOR: 1 A0116 / (BUILDQ: DB/TRIP (DESTINATION SDC)
                                     (TRAVELER A0114))
                  : T ; T))
```

Optimized interpretation:

```
(FOR: THE A0114 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (LASTNAME (QUOTE KLOVSTAD))
                         (FIRSTNAME (QUOTE JACK)))
      : T ; (FOR: 1 A0116 / (BUILD: DB/TRIP (DESTINATION (QUOTE SDC))
                                    (TRAVELER A0114))
                  : T ; T))
```

{This sentence is ambiguous. See other parsing on next page.}

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V SCHEDULE
     OBJ NP DET ART A
             N TRIP
             PP PREP FOR
                NPR NAME FIRST JACK
                         LAST KLOVSTAD
             PP PREP TO
                NP N SDC
                   FEATS NU SG
             FEATS NU SG
```

Interpretation:

```
(FOR: THE A0114 / (FINDQ: PERSON (LASTNAME KLOVSTAD)
                                 (FIRSTNAME JACK))
      : T ; (FOR: THE A0115 / (FINDQ: DB/CONFERENCE (LOCATION SDC))
                  : T ; (FOR: 1 A0117 / (BUILDQ: DB/TRIP
                                                 (TO/ATTEND A0115)
                                                 (TRAVELER A0114))
                                      : T ; T)))
```

Optimized interpretation:

```
(FOR: THE A0114 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (LASTNAME (QUOTE KLOVSTAD))
                         (FIRSTNAME (QUOTE JACK)))
      : T ; (FOR: THE A0115 / (FIND: (INSTANCE/OF (QUOTE DB/CONFERENCE)
                                     (LOCATION (QUOTE SDC)))
                  : T ; (FOR: 1 A0117 / (BUILD: DB/TRIP (TO/ATTEND A011
                                                 (TRAVELER A0114))
                                      : T ; T)))
```

(This is ambiguous because SDC can be either a destination or the sponsor of a meeting.)

```
**********************************************************
*                                                        *
* LYN -S TRIP TO FRANCE COST-PAST SIX HUNDRED DOLLAR-S   *
*                                                        *
**********************************************************
```

Found complete parse·

```
S DCL
  SUBJ NP DET ART THE
          N TRIP
          PP PREP FOR
             NP NPR NAME FIRST LYN
          PP PREP TO
             NP NPR LOCATION COUNTRY FRANCE
          FEATS NU SG
  AUX TNS PAST
      VOICE ACTIVE
  VP V COST
     OBJ NP $ 600
```

Interpretation:

```
[FOR: THE A0148 / (FINDQ: PERSON (FIRSTNAME LYN))
      : T ; (FOR: THE A0149 / (FINDQ: LOCATION (COUNTRY FRANCE))
                  : T ; (FOR: THE A0150 / (FINDQ: DB/TRIP
                                                  (DESTINATION A0149)
                                                  (TIME (BEFORE NOW))
                                                  (TRAVELER A0148))
                                : T ; (PROGN (ADD: A0150 VALUE 600)
                                             (ADD: A0150 UNITS DOLLARS]
```

Optimized interpretation:

```
[FOR: THE A0148 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (FIRSTNAME (QUOTE LYN)))
      : T ; (FOR: THE A0149 / (FINDLOC (INSTANCE/OF (QUOTE LOCATION))
                                       (COUNTRY (QUOTE FRANCE)))
                  : T ; (FOR: THE A0150 / (FIND: (INSTANCE/OF
                                                   (QUOTE DB/TRIP))
                                                 (DESTINATION A0149)
                                                 (TIME (QUOTE PAST))
                                                 (TRAVELER A0148))
                                : T ; (PROGN (ADD: A0150 (QUOTE VALUE)
                                                 (QUOTE 600))
                                             (ADD: A0150 (QUOTE UNITS)
                                                 (QUOTE DOLLARS]
```

{It is assumed that the statement is giving the system new information which must be added to the data base. Since only one form is allowed as an action, the LISP form PROGN which evaluates a sequence of forms must be used.}

```
**********************************************************
*                                                        *
* HOW MUCH IS IN THE SPEECH UNDERSTANDING BUDGET         *
*                                                        *
**********************************************************
```

Found complete parse:

```
S Q
  SUBJ NP DET QUANT HOWMUCH
          N ?
          FEATS NU MASS
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V BE
     PP PREP IN
        NP DET ART THE
           SPEECH UNDERSTANDING
           N BUDGET
           FEATS NU SG
```

Interpretation:

```
(FOR: THE A0146 / (FINDQ: DB/BUDGET (PROJECT (SPEECH UNDERSTANDING)))
      : T ; (OUTPUT: (GET: A0146 MONEY/REMAINING)))
```

Optimized interpretation:

```
[FOR: THE A0146 / [FIND: (INSTANCE/OF (QUOTE DB/BUDGET))
                         (PROJECT (QUOTE (SPEECH UNDERSTANDING]
      : T ; (OUTPUT: (GET: A0146 (QUOTE MONEY/REMAINING]
```

{Syntactically, the elliptical head of "how much" could be any mass noun, but in our environment the only semantic head is "money".}

```
**********************************************************************
*                                                                    *
* PLEASE GIVE ME THE ACTUAL COST OF TRIP NUMBER SIX EIGHT SEVEN THREE *
*                                                                    *
**********************************************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V GIVE
     OBJ NP DET ART THE
            ADJ ACTUAL
            N COST
            PP PREP OF
               6873
            FEATS NU SG
```

Interpretation:

```
(FOR: THE A0141 / (FINDQ: DB/TRIP (TRIP# 6873))
      : T ; (FOR: THE A0142 / (FINDQ: DB/COST (COST/OF A0141))
                  : T ; (OUTPUT: A0142)))
```

Optimized interpretation:

```
(FOR: THE A0141 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                         (TRIP# 6873))
      : T ; (FOR: THE A0142 / (FIND: (INSTANCE/OF (QUOTE DB/COST))
                                     (COST/OF A0141))
                  : T ; (OUTPUT: A0142)))
```

{Since there is currently no difference in the data base between "estimated cost" and "actal cost", the adjective has no effect on the interpretation which is constructed. The number given as a trip number must have the correct number of digits.}

```
**********************************************************************
*                                                                    *
* PLEASE SHOW ME JOHN MAKHOUL -S THREE TRIP-S TO PITTSBURGH          *
*                                                                    *
**********************************************************************
```

Found complete parse:

```
S IMP
  SUBJ NP PRO YOU
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V SHOW
     OBJ NP DET THE
             ADJ 3
             N TRIP
             PP PREP FOR
                NP NPR NAME FIRST JOHN
                            LAST MAKHOUL
             PP PREP TO
                NP NPR LOCATION CITY PITTSBURGH
             FEATS NU PL
```

Interpretation:

```
[FOR: THE A0131 / (FINDQ: PERSON (LASTNAME MAKHOUL)
                          (FIRSTNAME JOHN))
      : T ; (FOR: THE A0132 / (FINDQ: LOCATION (CITY PITTSBURGH))
                  : T ; (FOR: (THE 3)
                              A0133 / (FINDQ: DB/TRIP (DESTINATION A0132)
                                              (TRAVELER A0131))
                              : T ; (OUTPUT: A0133]
```

Optimized interpretation:

```
[FOR: THE A0131 / (FIND: (INSTANCE/OF (QUOTE PERSON))
                         (LASTNAME (QUOTE MAKHOUL))
                         (FIRSTNAME (QUOTE JOHN)))
      : T ; (FOR: THE A0132 / (FINDLOC (INSTANCE/OF (QUOTE LOCATION))
                                       (CITY (QUOTE PITTSBURGH)))
                  : T ; (FOR: (THE 3)
                              A0133 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                                             (DESTINATION A0132)
                                             (TRAVELER A0131))
                              : T ; (OUTPUT: A0133]
```

{This is straightforward except for the quantifier "three". Since the sentence implies that there are exactly 3 such trips, the semantic quantifier (THE 3) is generated.}

```
**************************
*                        *
* WHO IS GO-ING TO IFIP  *
*                        *
**************************
```

```
Found complete parse:
     S Q
       SUBJ NP PRO WHO
               FEATS NU SG/PL
       AUX TNS PRESENT PROGRESSIVE
           VOICE ACTIVE
       VP V GO
          PP PREP TO
             NP N IFIP
                FEATS NU SG
```

```
Interpretation:

     [FOR: THE A0014 / (FINDQ: DB/CONFERENCE (SPONSOR IFIP))
           : T ; (FOR: ALL A0015 / (FINDQ: DB/TRIP (TO/ATTEND A0014))
                     : T ; (OUTPUT: (GET: A0015 TRAVELER]
```

```
Optimized interpretation:

     [FOR: THE A0014 / (FIND: (INSTANCE/OF (QUOTE DB/CONFERENCE))
                              (SPONSOR (QUOTE IFIP)))
           : T ; (FOR: ALL A0015 / (FIND: (INSTANCE/OF (QUOTE DB/TRIP))
                                          (TO/ATTEND A0014))
                     : T ; (OUTPUT: (GET: A0015 (QUOTE TRAVELER]
```

{The first word causes the action to be set; the rest of the pause builds the information necessary to describe a trip.}

Appendix 3 - Annotated Syntax Trace

Appendix 3 contains an actual listing of a syntax trace file with explanatory notes added. The sentence was entered in text mode, one event at a time to simulate bi-directional middle-out parsing. While the sentence was run using BIGGRAM, and BIGDICT, all the relevant arcs used in its parsing are also in MIDGRAM. Thus, the interested reader may follow the trace in more detail by referring to Appendix 1. Page references within this appendix are to the page number at the top of the trace listing.

```
**************************************************
*                                                *
* BILL -S TRIP TO WASHINGTON COSTS NINETY DOLLAR-S *
*                                                *
**************************************************
```

This is a syntax trace for the above sentence, entered in text mode but simulating middle-out bi-directional parsing.

```
********
*      *
* TRIP *
*      *
********
```

Word match number

Word boundaries. These are arbitrary for text input

Word match lexical score (always 0 for text input)

Seed event

```
>Beginning new island with word match (3 TRIP 2 3 0)
```

Brief arc description (see trace listing page 1:1 for a more detailed explanation)

```
Doing arc: (WRD (TRIP)  -- (from S/THE 311) (to S/THE-TRIP 313))
     adding to left of PREMODS to get ((ADJ TRIP))

Creating SCONFIG group:
1:S/THE(2)-S/THE-TRIP(3)
Reqs = (PREMODS ((ADJ TRIP)))
States = S/THE
Scopedacts = NIL
Parent SC = NIL
```

Segment configuration. (see trace listing page 1:2 for detailed description)

```
Doing arc: (WRD (TRIP)  -- (from T-DET-SG-DEF/ 363) (to TRIP/NUMBER1 422
))

Creating SCONFIG group:
2:T-DET-SG-DEF/(2)-TRIP/NUMBER1(3)
Reqs = NIL
States = T-DET-SG-DEF/
Scopedacts = NIL
Parent SC = NIL

Doing arc: (WRD (TRIP)  -- (from T-SG/DET 377) (to TRIP/HEAD 420))
     setting SEMHEAD to DB/TRIP
     setting HEAD to (N TRIP)
     setting NU to SG
Saving scoped action 1 for Sconfig 3
     with scope T-DET-SG-DEF/ T-DET-SG-INDEF/ T-DETQ-SG/ T-ENTER/
T-SG-NON-DEM/ T-SG-POSS?/
Saving scoped action 2 for Sconfig 3
     with scope T-DET-SG-DEF/ T-DET-SG-INDEF/ T-DETQ-SG/ T-ENTER/
T-SG-NON-DEM/
```

```
Creating SCONFIG group:
3:T-SG/DET(2)-TRIP/HEAD(3)
Reqs = (NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-SG/DET
Scopedacts = (1 2)
Parent SC = NIL
Doing arc: (WRD (TRIP) -- (from TO/TAKE-A 405) (to TO/GO 393))
    setting OBJ to (NP (DET (ART A)) (N TRIP) (FEATS (NU SG)))
Creating SCONFIG group:
4:TO/TAKE-A(2)-TO/GO(3)
Reqs = (OBJ (NP (DET (ART A)) (N TRIP) (FEATS (NU SG))))
States = TO/TAKE-A
Scopedacts = NIL
Parent SC = NIL
```

Arc type

Arc label

Left state name and number

Right state name and number

Arc action setting a register

```
Doing left proposals for Iconfig 1
     found 39 arcs reaching Iconfig.
Doing right proposals for Iconfig 1
     found 51 arcs reaching Iconfig.
```

These refer to the number of different terminal-consuming arcs from which the predictions for the island are drawn.

```
Created new island:

Island config 1 = 2 TRIP 3
Left spansflag = NIL
Right spansflag - NIL
Parent IC = NIL
```

Island configuration.

Spanning flags indicate whether the end of the utterance has been reached. Referred to as "[" or "]" in the boxes.

Adding possessive morpheme -S to left of island containing the word TRIP.

```
**************
*            *
* -S <- TRIP *
*            *
**************
```

```
Processing event for Iconfig 1 with word match (2 -S 1 2 0)
Spanning flags  Left: NIL   Right: NIL

Extending To Do Queue from Sc's: (1 2 3 4)
```

These are the SCONFIGs from the old island.

```
Doing consuming arcs for the event.
```

The fact that no arcs are processed between extending the TODOQ and doing the consuming arcs, indicates that there are no intervening levels or arcs between the old island and the new arc.

```
Doing arc: (WRD (-S)  -- (from T-SG-POSS?/ 376)  (to T-SG/DET 377))
Saving scoped action 3 for Sconfig 5
     with scope T-SG-NON-DEM/
Doing scoped action 1 for SC 5
     in arc (WRD (TRIP)  -- (from T-SG/DET 377)  (to TRIP/HEAD 420))

Creating SCONFIG group:
5:T-SG-POSS?/(1)-TRIP/HEAD(3)
Reqs = (NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-SG-POSS?/ T-SG/DET
Scopedacts = (3 2)
Parent SC = 3

Doing left proposals for Iconfig 2
     found 3 arcs reaching Iconfig.
Doing right proposals for Iconfig 2
     found 42 arcs reaching Iconfig.

Created new island:

Island config 2 = 1 -S 2 TRIP 3
Left spansflag = NIL
Right spansflag - NIL
Parent IC = 1
```

Saving action to wait for proper left context, see page 1:5 for execution.

<SCONFIG number>:
<Left state>(<Left boundary>)-
<Right state>(<Right boundary>)

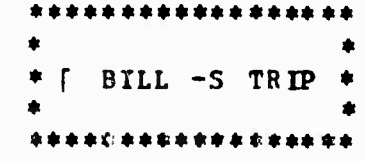
Current register settings

Left states of arcs already executed

Outstanding scoped actions

```
**********************
*                    *
* BILL <- -S TRIP    *
*                    *
**********************

Processing event for Iconfig 2 with word match (1 BILL 0 1 0)
Spanning flags  Left: NIL   Right: NIL


Extending To Do Queue from Sc's: (5)

Extending UC!Q: (5)

Creating new SCONFIG group at different level:
6:NAME/POP(1)-NAME/POP(1)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL
```

```
Doing arc: (JUMP NAME/POP  -- (from NAME/FIRST 124) (to NAME/POP 125))
Saving scoped action 4 for Sconfig 7
     with scope NAME/

Creating SCONFIG group:
7:NAME/FIRST(1)-NAME/POP(1)
Regs = NIL
States = NAME/FIRST
Scopedacts = (4)
Parent SC = 6

Doing consuming arcs for the event.


Doing arc: (CAT FIRSTNAME  -- (from NAME/ 123) (to NAME/FIRST 124))
     setting NAME to T
     setting FIRST to BILL
Doing scoped action 4 for SC 8
     in arc (JUMP NAME/POP  -- (from NAME/FIRST 124) (to NAME/POP 125))

Creating SCONFIG group:
8:NAME/(0)-NAME/POP(1)
Regs = (FIRST BILL NAME T)
States = NAME/ NAME/FIRST
Scopedacts = NIL
Parent SC = 7

Doing left proposals for Iconfig 3
     found 27 arcs reaching Iconfig.
Doing right proposals for Iconfig 3
     found 42 arcs reaching Iconfig.

Created new island:

Island config 3 = 0 BILL 1 -S 2 TRIP 3
Left spansflag = NIL
Right spansflag - NIL
Parent IC = 2
```



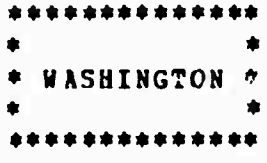
Doing end event for the left end of the island, i.e., connecting the island to the start state, S/.

```
Trying to connect to left end of sentence.

Extending To Do Queue from Sc's: (8)

Splitting the UB!Queue: (8)

Completing constituents for the UB!COMPLETEQ: (8)

Doing arc: (POP  --  (from NAME/POP 125))
     setting INTERP to (! THE A0013 PERSON ((FIRSTNAME BILL)))
     setting #LIFTLIST# to (INTERP)
     lifting  register INTERP.
Saving scoped action 5 for Sconfig 9
     with scope POP

Creating SCONFIG group:
9:NAME/(0)-NAME/POP(1)
Reqs = (#LIFTLIST# (INTERP) INTERP (! THE A0013 PERSON ((FIRSTNAME
BILL))) FIRST BILL NAME T)
States = NAME/POP NAME/ NAME/FIRST
Scopedacts = (5)
Parent SC = 8

Creating direct connection to new constituent from:

5:T-SG-POSS?/(1)-TRIP/HEAD(3)
Reqs = (NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-SG-POSS?/ T-SG/DET
Scopedacts = (3 2)
Parent SC = 3


Creating constituent from Sconfig 9:

(! THE A0013 PERSON ((FIRSTNAME BILL)))<----Semantic representation of
                                            new constituent

NPR NAME FIRST BILL<----------Syntactic tree representation of
                              new constituent
```

```
Doing arc: (PUSH NAME/  -- (from T-SG-NON-DEM/ 375) (to T-SG-POSS?/ 376
))
     setting SEMQUANT to THE
     adding to left of SEMLINKS to get ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL)))))
     setting DET to (DET THE)
     setting PERSONMOD to T
     setting PERSON to (NPR (NAME (FIRST BILL)))
Doing scoped action 3 for SC 10
     in arc (WRD (-S)  -- (from T-SG-POSS?/ 376) (to T-SG/DET 377))
     setting DET to (DET (ART THE))
     adding to left of NMODS to get ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL))))))
Doing scoped action 2 for SC 10
     in arc (WRD (TRIP)  -- (from T-SG/DET 377)  (to TRIP/HEAD 420))

Creating SCONFIG group:
10:T-SG-NON-DEM/(-)-TRIP/HEAD(3)
Reqs = (NMODS ((PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) DET
(DET (ART THE)) PERSON (NPR (NAME (FIRST BILL))) PERSONMOD T DET (DET
THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 5

Extending To Do Queue from Sc's: (10)

Doing arc: (JUMP T-SG-NON-DEM/  -- (from T-DET-SG-DEF/ 363) (to
T-SG-NON-DEM/ 375))

Creating SCONFIG group:
11:T-DET-SG-DEF/(0)-TRIP/HEAD(3)
Reqs = (NMODS ((PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) DET
(DET (ART THE)) PERSON (NPR (NAME (FIRST BILL))) PERSONMOD T DET (DET
THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 10

Doing arc: (JUMP T-DET-SG-DEF/  -- (from T-DET-SG/ 365) (to
T-DET-SG-DEF/ 363))

Creating SCONFIG group:
12:T-DET-SG/(0)-TRIP/HEAD(3)
Reqs = (NMODS ((PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) DET
(DET (ART THE)) PERSON (NPR (NAME (FIRST BILL))) PERSONMOD T DET (DET
THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = T-DET-SG/ T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 11
```

Execution of scoped action saved on page 1:2 above. Note that scope is same as left state of the arc being executed here.

```
Splitting the UB!Queue: (12)

Extending UB!OPENRIGHTQ: (12)

Creating new SCONFIG group at different level:
13:START/(0)-START/(-)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL

Doing arc: (JUMP START/  -- (from S/ 155) (to START/ 357))

Creating SCONFIG group:
14:S/(0)-START/(-)
Reqs = NIL
States = S/
Scopedacts = NIL
Parent SC = 13

Doing left proposals for Iconfig 4
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 4
     found 18 arcs reaching Iconfig.

Created new island:

Island config 4 = 0 BILL 1 -S 2 TRIP 3
Left spansflag = T
Right spansflag - NIL
Parent IC = 3
```



```
                                        Beginning new seed.
Beginning new island with word match (5 WASHINGTON 4 5 0)


Doing arc: (CAT CITY  -- (from CITY/ 39) (to CITY/CITY 40))
     adding to left of SEMLINKS to get ((CITY WASHINGTON))
     setting CITY to WASHINGTON
     setting LOCATION to T

Creating SCONFIG group:
15:CITY/(4)-CITY/CITY(5)
Reqs = (LOCATION T CITY WASHINGTON SEMLINKS ((CITY WASHINGTON)))
States = CITY/
Scopedacts = NIL
Parent SC = NIL

Doing arc: (CAT STATE  -- (from CITY/ 39) (to CITY/POP 41))
     setting LOCATION to T
     setting AMBIG to WASHINGTON
     adding to left of SEMLINKS to get ((AMBIG WASHINGTON))

Creating SCONFIG group:
16:CITY/(4)-CITY/POP(5)
Reqs = (SEMLINKS ((AMBIG WASHINGTON)) AMBIG WASHINGTON LOCATION T)
States = CITY/
Scopedacts = NIL
Parent SC = NIL

Doing arc: (CAT STATE  -- (from CITY/CITY 40) (to CITY/POP 41))
     adding to left of SEMLINKS to get ((STATE WASHINGTON))
     setting STATE to WASHINGTON
Saving scoped action 6 for Sconfig 17
     with scope CITY/

Creating SCONFIG group:
17:CITY/CITY(4)-CITY/POP(5)
Reqs = (STATE WASHINGTON SEMLINKS ((STATE WASHINGTON)))
States = CITY/CITY
Scopedacts = (6)
Parent SC = NIL

Doing left proposals for Iconfig 5
     found 19 arcs reaching Iconfig.
Doing right proposals for Iconfig 5
     found 50 arcs reaching Iconfig.

Created new island:

Island config 5 = 4 WASHINGTON 5
Left spansflag = NIL
Right spansflag - NIL
Parent IC = NIL
```

```
**************************************
*                                    *
* [ BILL -S TRIP -> TO <- WASHINGTON *
*                                    *
**************************************
```

Island collision event, adding new word TO. Processing extends longer of the two islands by adding the new word, then adding the words from the shorter island one at a time.

```
Processing event for Iconfig 4 with word match (4 TO 3 4 0)
Spanning flags  Left: T   Right: NIL


Extending To Do Queue from Sc's: (12)

Extending B!Q: (12)

Creating new SCONFIG group at different level:
18:TRIP-TO/(3)-TRIP-TO/(3)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL

Doing consuming arcs for the event.


Doing arc: (WRD (TO)  -- (from TRIP-TO/ 410) (to TRIP/TO 427))

Creating SCONFIG group:
19:TRIP-TO/(3)-TRIP/TO(4)
Reqs = NIL
States = TRIP-TO/
Scopedacts = NIL
Parent SC = 18

Doing left proposals for Iconfig 6
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 6
     found 9 arcs reaching Iconfig.

Created new island:

Island config 6 = 0 BILL 1 -S 2 TRIP 3 TO 4
Left spansflag = T
Right spansflag - NIL
Parent IC = 4
```

Island configuration created as intermediate step of island collision processing.

This is part of island collision processing.

```
Processing event for Iconfig 6 with word match (5 WASHINGTON 4 5 0)
Spanning flags  Left: T   Right: NIL
```

```
Extending To Do Queue from Sc's: (19)

Extending B!Q: (19)

Creating new SCONFIG group at different level:
20:CITY/(4)-CITY/(4)
Regs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL

Doing consuming arcs for the event.


Doing arc: (CAT CITY  -- (from CITY/ 39) (to CITY/CITY 40))
     adding to left of SEMLINKS to get ((CITY WASHINGTON))
     setting CITY to WASHINGTON
     setting LOCATION to T

Creating SCONFIG group:
21:CITY/(4)-CITY/CITY(5)
Regs = (LOCATION T CITY WASHINGTON SEMLINKS ((CITY WASHINGTON)))
States = CITY/
Scopedacts = NIL
Parent SC = 20

Doing arc: (CAT STATE  -- (from CITY/ 39) (to CITY/POP 41))
     setting LOCATION to T
     setting AMBIG to WASHINGTON
     adding to left of SEMLINKS to get ((AMBIG WASHINGTON))

Creating SCONFIG group:
22:CITY/(4)-CITY/POP(5)
Regs = (SEMLINKS ((AMBIG WASHINGTON)) AMBIG WASHINGTON LOCATION T)
States = CITY/
Scopedacts = NIL
Parent SC = 20

Doing left proposals for Iconfig 7
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 7
     found 17 arcs reaching Iconfig.

Created new island:

Island config 7 = 0 BILL 1 -S 2 TRIP 3 TO 4 WASHINGTON 5
Left spansflag = T
Right spansflag - NIL
Parent IC = 6
```

```
*****************************************
*                                       *
* [ BILL -S TRIP TO WASHINGTON -> COSTS *
*                                       *
*****************************************
```

Note that this event requires the completion of 3 lower level constituents.

```
Processing event for Iconfig 7 with word match (6 COSTS 5 6 0)
Spanning flags  Left: T   Right: NIL


Extending To Do Queue from Sc's: (22 21)

Doing arc: (JUMP CITY/POP  -- (from CITY/CITY 40) (to CITY/POP 41))
Verify failed for action. (NOT (CATCHECK (WORD (GETR CITY)) (QUOTE
STATE)))
  in arc: (JUMP CITY/POP  -- (from CITY/CITY 40) (to CITY/POP 41))
Arc failed.

Splitting the C!Queue: (22)

Completing constituents for the C!COMPLETEQ: (22)

Doing arc: (POP  -- (from CITY/POP 41))
     setting INTERP to (! THE A0014 LOCATION ((AMBIG WASHINGTON)))
     setting #LIFTLIST# to (INTERP)
     lifting  register INTERP.
Saving scoped action 7 for Sconfig 23
     with scope POP

Creating SCONFIG group:
23:CITY/(4)-CITY/POP(5)
Reqs = (#LIFTLIST# (INTERP) INTERP (! THE A0014 LOCATION ((AMBIG
WASHINGTON))) SEMLINKS ((AMBIG WASHINGTON)) AMBIG WASHINGTON LOCATION
T)
States = CITY/POP CITY/
Scopedacts = (7)
Parent SC = 22

Creating direct connection to new constituent from:

19:TRIP-TO/(3)-TRIP/TO(4)
Reqs = NIL
States = TRIP-TO/
Scopedacts = NIL
Parent SC = 18


Creating constituent from Sconfig 23:

(! THE A0014 LOCATION ((AMBIG WASHINGTON)))

NPR LOCATION AMBIG WASHINGTON
```

```
Doing arc: (PUSH CITY/  -- (from TRIP/TO 427) (to TRIP-TO/POP 411))
     setting INTERP to (DESTINATION (! THE A0014 LOCATION ((AMBIG
WASHINGTON))))
     setting PP to (PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))

Creating SCONFIG group:
24:TRIP-TO/(3)-TRIP-TO/POP(6)
Reqs = (PP (PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
INTERP (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON)))))
States = TRIP/TO TRIP-TO/
Scopedacts = NIL
Parent SC = 19

Extending To Do Queue from Sc's: (24)

Splitting the C!Queue: (24)

Completing constituents for the C!COMPLETEQ: (24)

Doing arc: (POP  -- (from TRIP-TO/POP 411))
     setting #LIFTLIST# to (INTERP)
     lifting  register INTERP.
     setting #LIFTLIST# to (SEMLINKS INTERP)
     lifting  register SEMLINKS.
Saving scoped action 8 for Sconfig 25
     with scope POP

Creating SCONFIG group:
25:TRIP-TO/(3)-TRIP-TO/POP(5)
Reqs = (#LIFTLIST# (SEMLINKS INTERP) #LIFTLIST# (INTERP) PP (PP (PREP
TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))) INTERP (DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))))
States = TRIP-TO/POP TRIP/TO TRIP-TO/
Scopedacts = (8)
Parent SC = 24

Creating direct connection to new constituent from:

12:T-DET-SG/(0)-TRIP/HEAD(3)
Reqs = (NMODS ((PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) DET
(DET (ART THE)) PERSON (NPR (NAME (FIRST BILL))) PERSONMOD T DET (DET
THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD CB/TRIP)
States = T-DET-SG/ T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 11


Creating constituent from Sconfig 25:

(DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))

PP PREP TO
   NP NPR LOCATION AMBIG WASHINGTON
```

```
Doing arc: (PUSH TRIP-TO/ -- (from TRIP/HEAD 420) (to TRIP/TO-MOD 429))
     adding to left of SEMLINKS to get ((DESTINATION (! THE A0014
LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL)))))
     adding to left of NMODS to get ((PP (PREP TO) (NP (NPR (LOCATION
(AMBIG WASHINGTON))))) (PP (PREP FOR) (NP (NPR (NAME (FIRST BILL))))))

Creating SCONFIG group:
26:T-DET-SG/(0)-TRIP/TO-MOD(6)
Reqs = (NMODS ((PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
(PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) SEMLINKS ((DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL))))) NMODS ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL)))))) DET (DET (ART THE)) PERSON (NFR (NAME (FIRST BILL)))
PERSONMOD T DET (DET THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL)))}) SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = TRIP/HEAD T-DET-SG/ T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/
T-SG/DET
Scopedacts = NIL
Parent SC = 12

Extending To Do Queue from Sc's: (26)

Doing arc: (JUMP TRIP/TO-FOR-MOD -- (from TRIP/TO-MOD 429) (to
TRIP/TO-FOR-MOD 428))

Creating SCONFIG group:
27:T-DET-SG/(0)-TRIP/TO-FOR-MOD(5)
Reqs = (NMODS ((PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
(PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) SEMLINKS ((DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL))))) NMODS ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL)))))) DET (DET (ART THE)) PERSON (NFR (NAME (FIRST BILL)))
PERSONMOD T DET (DET THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL))))) SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = TRIP/TO-MOD TRIP/HEAD T-DET-SG/ T-DET-SG-DEF/ T-SG-NON-DEM/
T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 26

Doing arc: (JUMP TRIP/DATE? -- (from TRIP/TO-FOR-MOD 428) (to
TRIP/DATE? 416))

Creating SCONFIG group:
28:T-DET-SG/(0)-TRIP/DATE?(5)
Reqs = (NMODS ((PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
(PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) SEMLINKS ((DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL))))) NMODS ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL)))))) DET (DET (ART THE)) PERSON (NFR (NAME (FIRST BILL)))
PERSONMOD T DET (DET THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL))))) SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = TRIP/TO-FOR-MOD TRIP/TO-MOD TRIP/HEAD T-DET-SG/ T-DET-SG-DEF/
T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 27
```

```
Doing arc: (JUMP TRIP/POP -- (from TRIP/DATE? 416) (to TRIP/POP 425))

Creating SCONFIG group:
29:T-DET-SG/(0)-TRIP/POP(5)
Reqs = (NMODS ((PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))))
(PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) SEMLINKS ((DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL))))) NMODS ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL)))))) DET (DET (ART THE)) PERSON (NPR (NAME (FIRST BILL)))
PERSONMOD T DET (DET THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL))))) SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = TRIP/DATE? TRIP/TO-FOR-MOD TRIP/TO-MOD TRIP/HEAD T-DET-SG/
T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = NIL
Parent SC = 28

Splitting the C!Queue: (29)

Completing constituents for the C!COMPLETEQ: (29)

Doing arc: (POP -- (from TRIP/POP 425))
Saving scoped action 9 for Sconfig 30
     with scope T
Saving scoped action 10 for Sconfig 30
     with scope POP

Creating SCONFIG group:
30:T-DET-SG/(0)-TRIP/POP(5)
Reqs = (NMODS ((PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
(PP (PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) SEMLINKS ((DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL))))) NMODS ((PP (PREP FOR) (NP (NPR (NAME
(FIRST BILL)))))) DET (DET (ART THE)) PERSON (NPR (NAME (FIRST BILL)))
PERSONMOD T DET (DET THE) SEMLINKS ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL))))) SEMQUANT THE NU SG HEAD (N TRIP) SEMHEAD DB/TRIP)
States = TRIP/POP TRIP/DATE? TRIP/TO-FOR-MOD TRIP/TO-MOD TRIP/HEAD
T-DET-SG/ T-DET-SG-DEF/ T-SG-NON-DEM/ T-SG-POSS?/ T-SG/DET
Scopedacts = (9 10)
Parent SC = 29
Doing scoped action 9 for SC 30
     in arc (POP -- (from TRIP/POP 425))
     setting SEMLINKS to ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME
BILL)))) (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON)))))
     setting INTERP to (! THE A0015 DB/TRIP ((TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL)))) (DESTINATION (! THE A0014 LOCATION ((AMBIG
WASHINGTON))))))
     setting #LIFTLIST# to (INTERP)
     lifting register INTERP.

Creating direct connection to new constituent from:

14:S/(0)-START/(-)
Reqs = NIL
States = S/
Scopedacts = NIL
Parent SC = 13
```

```
;  <GBROWN>APPENDIX-TRACE.SYNTAX;4    Mon 20-Dec-76 6:12PM  PAGE 1:14

Creating constituent from Sconfig 30:

(! THE A0015 DB/TRIP ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
    (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))))

NP DET ART THE
   N TRIP
   PP PREP FOR
      NP NPR NAME FIRST BILL
   PP PREP TO
      NP NPR LOCATION AMBIG WASHINGTON
   FEATS NU SG


Doing arc: (PUSH T-DET-SG/  -- (from START/ 357) (to S/DCL-SUBJ-SG 203))
     setting SEMQUANT to THE
     setting SEMVAR to A0015
     setting SEMHEAD to DB/TRIP
     adding to left of SEMLINKS to get ((TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL)))))
     adding to left of SEMLINKS to get ((DESTINATION (! THE A0014
LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL)))))
     setting TYPE to DCL
     setting SUBJ to (NP (DET (ART THE)) (N TRIP) (PP (PREP FOR) (NP
(NPR (NAME (FIRST BILL))))) (PP (PREP TO) (NP (NPR (LOCATION (AMBIG
WASHINGTON))))) (FEATS (NU SG)))

Creating SCONFIG group:
31:S/(0)-S/DCL-SUBJ-SG(6)
Reqs = (SUBJ (NP (DET (ART THE)) (N TRIP) (PP (PREP FOR) (NP (NPR
(NAME (FIRST BILL))))) (PP (PREP TO) (NP (NPR (LOCATION (AMBIG
WASHINGTON))))) (FEATS (NU SG))) TYPE DCL SEMLINKS ((DESTINATION (!
THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON
((FIRSTNAME BILL))))) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL))))) SEMHEAD DB/TRIP SEMVAR A0015 SEMQUANT THE)
States = START/ S/
Scopedacts = NIL
Parent SC = 14

Extending To Do Queue from Sc's: (31)

Doing consuming arcs for the event.
```

Semantic representation of "BILL'S TRIP TO WASHINGTON".

Syntactic tree for "BILL'S TRIP TO WASHINGTON".

```
;  <GBROWN>APPENDIX-TRACE.SYNTAX;4    Mon 20-Dec-76 6:12PM  PAGE 1:15

Doing arc: (WRD (IS WAS COSTS COST-PAST)  -- (from S/DCL-SUBJ-SG 203
) (to S/DCL-IS 200))
     setting V to COST
     setting HEAD to COST
     setting TNS to PRESENT
     setting VOICE to ACTIVE
     adding to left of SEMLINKS to get ((TIME (AFTER NOW)) (DESTINATION
(! THE A0014 LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL)))))

Creating SCONFIG group:
32:S/(0)-S/DCL-IS(6)
Reqs = (SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ (NP (DET (ART THE))
(N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST BILL))))) (PP (PREP
TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))) (BEATS (NU SG))) TYPE
DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
(TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMLINKS ((TRAVELER
(! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD DB/TRIP SEMVAR A0015
SEMQUANT THE)
States = S/DCL-SUBJ-SG START/ S/
Scopedacts = NIL
Parent SC = 31

Doing left proposals for Iconfig 8
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 8
     found 8 arcs reaching Iconfig.

Created new island:

Island config 8 = 0 BILL 1 -S 2 TRIP 3 TO 4 WASHINGTON 5 COSTS 6
Left spansflag = T
Right spansflag - NIL
Parent IC = 7
```

```
**************************************************
*                                                *
* [ BILL -S TRIP TO WASHINGTON COSTS -> NINETY *
*                                                *
**************************************************
```

```
: <GBROWN>APPENDIX-TRACE.SYNTAX;4    Mon 20-Dec-76 6:12PM  PAGE 1:16

Processing event for Iconfig 8 with word match (7 NINETY 6 7 0)
Spanning flags  Left: T   Right: NIL


Extending To Do Queue from Sc's: (32)

Doing arc: (JUMP S/DCL-IS-ADV  -- (from S/DCL-IS 200) (to S/DCL-IS-ADV
201))

Creating SCONFIG group:
33:S/(0)-S/DCL-IS-ADV(6)
Reqs = (SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ (NP (DET (ART THE))
(N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST BILL))))) (PP (PREP
TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))) (FEATS (NU SG))) TYPE
DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
(TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMLINKS ((TRAVELER
(! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD DB/TRIP SEMVAR A0015
SEMQUANT THE)
States = S/DCL-IS S/DCL-SUBJ-SG START/ S/
Scopedacts = NIL
Parent SC = 32

Extending BIQ: (33)

Creating new SCONFIG group at different level:
34:NUM/(6)-NUM/(6)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL

Doing consuming arcs for the event.


Doing arc: (CAT TENS  -- (from NUM/ 126) (to NUM/1 127))
     setting NUM to 90

Creating SCONFIG group:
35:NUM/(6)-NUM/1(7)
Reqs = (NUM 90)
States = NUM/
Scopedacts = NIL
Parent SC = 34

Doing left proposals for Iconfig 9
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 9
     found 5 arcs reaching Iconfig.

Created new island:
```

```
Island config 9 = 0 BILL 1 -S 2 TRIP 3 TO 4 WASHINGTON 5 COSTS 6 NINETY
7
Left spansflag = T
Right spansflag - NIL
Parent IC = 8



      ************************************************************
      *                                                          *
      * [ BILL -S TRIP TO WASHINGTON COSTS NINETY -> DOLLAR-S    *
      *                                                          *
      ************************************************************


Processing event for Iconfig 9 with word match (8 DOLLAR-S 7 6 0)
Spanning flags  Left: T   Right: NIL


Extending To Do Queue from Sc's: (35)

Doing arc: (JUMP NUM/NUM  -- (from NUM/1 127) (to NUM/NUM 128))

Creating SCONFIG group:
36:NUM/(6)-NUM/NUM(7)
Reqs = (NUM 90)
States = NUM/1 NUM/
Scopedacts = NIL
Parent SC = 35

Splitting the C!Queue: (36)

Completing constituents for the C!COMPLETEQ: (36)

Doing arc: (POP  -- (from NUM/NUM 128))
Saving scoped action 11 for Sconfig 37
     with scope POP

Creating SCONFIG group:
37:NUM/(6)-NUM/NUM(7)
Reqs = (NUM 90)
States = NUM/NUM NUM/1 NUM/
Scopedacts = (11)
Parent SC = 36
Creating SCONFIG group at new level for complete constituent
38:NUMBER/(7)-NUMBER/(7)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL
```

```
Creating constituent from Sconfig 37:

NIL                    Note that syntactic representation of this
                       constituent is sufficient here.

90


Doing arc: (PUSH NUM/  -- (from NUMBER/ 129) (to NUMBER/NUM 133))
     setting NUM to 90

Creating SCONFIG group:
39:NUMBER/(7)-NUMBER/NUM(8)
Reqs = (NUM 90)
States = NUMBER/
Scopedacts = NIL
Parent SC = 38

Extending To Do Queue from Sc's: (39)

Splitting the C!Queue: (39)

Completing constituents for the C!COMPLETEQ: (39)

Doing arc: (POP  -- (from NUMBER/NUM 133))
Saving scoped action 12 for Sconfig 40
     with scope POP

Creating SCONFIG group:
40:NUMBER/(7)-NUMBER/NUM(7)
Reqs = (NUM 90)
States = NUMBER/NUM NUMBER/
Scopedacts = (12)
Parent SC = 39
Creating SCONFIG group at new level for complete constituent
41:AMOUNT/(7)-AMOUNT/(7)
Reqs = NIL
States = NIL
Scopedacts = NIL
Parent SC = NIL


Doing arc: (JUMP AMOUNT/MOD  -- (from AMOUNT/ 1) (to AMOUNT/MOD 6))

Creating SCONFIG group:
42:AMOUNT/(7)-AMOUNT/MOD(7)
Reqs = NIL
States = AMOUNT/
Scopedacts = NIL
Parent SC = 41

Creating constituent from Sconfig 40:

NIL

90
```

```
Doing arc: (PUSH NUMBER/  -- (from AMOUNT/MOD 6) (to AMOUNT/NUM 7))
     setting NUM to 90

Creating SCONFIG group:
43:AMOUNT/(7)-AMOUNT/NUM(8)
Reqs = (NUM 90)
States = AMOUNT/MOD AMOUNT/
Scopedacts = NIL
Parent SC = 42

Extending To Do Queue from Sc's: (43)

Doing consuming arcs for the event.


Doing arc: (WRD (DOLLAR DOLLAR-S BUCKS)  -- (from AMOUNT/NUM 7) (to
AMOUNT/DOLLARS 5))
     setting NU to NIL

Creating SCONFIG group:
44:AMOUNT/(7)-AMOUNT/DOLLARS(8)
Reqs = (NU NIL NUM 90)
States = AMOUNT/NUM AMOUNT/MOD AMOUNT/
Scopedacts = NIL
Parent SC = 43

Doing left proposals for Iconfig 10
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 10
     found 3 arcs reaching Iconfig.

Created new island:

Island config 10 = 0 BILL 1 -S 2 TRIP 3 TO 4 WASHINGTON 5 COSTS 6 NINETY
 7 DOLLAR-S 8
Left spansflag = T
Right spansflag - NIL
Parent IC = 9
```

```
**********************************************************
*                                                        *
* [ BILL -S TRIP TO WASHINGTON COSTS NINETY DOLLAR-S -> ] *
*                                                        *
**********************************************************
```

Doing right end event.

```
Trying to connect to right end of sentence.

Extending To Do Queue from Sc's: (44)

Doing arc: (JUMP AMOUNT/POP -- (from AMOUNT/DOLLARS 5) (to AMOUNT/POP 8
))

Creating SCONFIG group:
45:AMOUNT/(7)-AMOUNT/POP(8)
Reqs = (NO NIL NUM 90)
States = AMOUNT/DOLLARS AMOUNT/NUM AMOUNT/MOD AMOUNT/
Scopedacts = NIL
Parent SC = 44

Splitting the C!Queue: (45)

Completing constituents for the C!COMPLETEQ: (45)

Doing arc: (POP -- (from AMOUNT/POP 8))
     setting AMT to 90
     setting #LIFTLIST# to (AMT)
     lifting  register AMT.
     setting #LIFTLIST# to (MOD AMT)
     lifting  register MOD.
Saving scoped action 13 for Sconfig 46
     with scope POP

Creating SCONFIG group:
46:AMOUNT/(7)-AMOUNT/POP(8)
Reqs = (#LIFTLIST# (MOD AMT) #LIFTLIST# (AMT) APT 90 NO NIL NUM 90)
States = AMOUNT/POP AMOUNT/DOLLARS AMOUNT/NUM AMOUNT/MOD AMOUNT/
Scopedacts = (13)
Parent SC = 45

Creating direct connection to new constituent from:

33:S/(0)-S/DCL-IS-ADV(6)
Reqs = (SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ (NP (DET (ART THE))
(N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST BILL))))) (PP (PREP
TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))) (FEATS (NU SG))) TYPE
DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
(TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMLINKS ((TRAVELER
(! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD DB/TRIP SEMVAR A0015
SEMQUANT THE)
States = S/DCL-IS S/DCL-SUBJ-SG START/ S/
Scopedacts = NIL
Parent SC = 32


Creating constituent from Sconfig 46:

NIL
```

```
$ 90


Doing arc: (PUSH AMOUNT/  -- (from S/DCL-IS-ADV 201) (to S/POP 276))
     setting ACTION to (PROGN (ADD: *** VALUE 90) (ADD: *** UNITS
DOLLARS))
     setting OBJ to (NP ($ 90))

Creating SCONFIG group:
47:S/(0)-S/POP(-)
Regs = (OBJ (NP ($ 90)) ACTION (PROGN (ADD: *** VALUE 90) (ADD: ***
UNITS DOLLARS)) SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014
LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL))))) VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ
(NP (DET (ART THE)) (N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST
BILL))))) (PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON)))))
(FEATS (NU SG))) TYPE DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD
DB/TRIP SEMVAR A0015 SEMQUANT THE)
States = S/DCL-IS-ADV S/DCL-IS S/DCL-SUBJ-SG START/ S/
Scopedacts = NIL
Parent SC = 33

Extending To Do Queue from Sc's: (47)

Doing arc: (JUMP END/  -- (from S/POP 276) (to END/ 78))
     setting SEMLINKS to ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME
BILL)))) (TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION ((AMBIG
WASHINGTON)))))
     setting INTERP to (! THE A0015 DB/TRIP ((TRAVELER (! THE A0013
PERSON ((FIRSTNAME BILL)))) (TIME (AFTER NOW)) (DESTINATION (! THE
A0014 LOCATION ((AMBIG WASHINGTON))))) NIL NIL)
     setting INTERP to (FOR: THE A0013 / (FINDQ: PERSON (FIRSTNAME
BILL)) : T ; (FOR: THE A0014 / (FINDQ: LOCATION (AMBIG WASHINGTON))
: T ; (FOR: THE A0015 / (FINDQ: DB/TRIP (DESTINATION A0014) (TIME
(AFTER NOW)) (TRAVELER A0013)) : T ; (PROGN (ADD: A0015 VALUE 90)
(ADD: A0015 UNITS DOLLARS)))))
     setting POPVAL to (S DCL (SUBJ (NP (DET (ART THE)) (N TRIP) (PP
(PREP FOR) (NP (NPR (NAME (FIRST BILL)))))) (PP (PREP TO) (NP (NPR
(LOCATION (AMBIG WASHINGTON))))) (FEATS (NU SG))) (AUX (TNS PRESENT)
(VOICE ACTIVE)) (VP (V COST) (OBJ (NP ($ 90))))))
```

```
Creating SCONFIG group:
48:S/(0)-END/(8)
Reqs = (POPVAL (S DCL (SUBJ (NP (DET (ART THE)) (N TRIP) (PP (PREP
FOR) (NP (NPR (NAME (FIRST BILL))))) (PP (PREP TO) (NP (NPR (LOCATION
(AMBIG WASHINGTON))))) (FEATS (NU SG)))) (AUX (TNS PRESENT) (VOICE
ACTIVE)) (VP (V COST) (OBJ (NP ($ 90))))) INTERP (FOR: THE A0013 /
(FINDQ: PERSON (FIRSTNAME BILL)) : T ; (FOR: THE A0014 / (FINDQ:
LOCATION (AMBIG WASHINGTON)) : T ; (FOR: THE A0015 / (FINDQ: DB/TRIP
(DESTINATION A0014) (TIME (AFTER NOW)) (TRAVELER A0013)) : T : (PROGN
(ADD: A0015 VALUE 90) (ADD: A0015 UNITS DOLLARS))))) INTERP (! THE
A0015 DB/TRIP ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
(TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
                                        **)
NIL NIL) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
(TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
                                        **)
OBJ (NP ($ 90)) ACTION (PROGN (ADD: *** VALUE 90) (ADD: *** UNITS
DOLLARS)) SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014
LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL))))) VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ
(NP (DET (ART THE)) (N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST
BILL)))))) (PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))))
(FEATS (NU SG))) TYPE DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD
DB/TRIP SEMVAR A0015 SEMQUANT THE)
States = S/POP S/DCL-IS-ADV S/DCL-IS S/DCL-SUBJ-SG START/ S/
Scopedacts = NIL
Parent SC = 47

Doing consuming arcs for the event.


Doing arc: (POP -- (from END/ 78))
Saving scoped action 14 for Sconfig 49
     with scope POP

Creating SCONFIG group:

49:S/(0)-END/(-)
FOR) (NP (NPR (NAME (FIRST BILL))))) (PP (PREP TO) (NP (NPR (LOCATION
(AMBIG WASHINGTON))))) (FEATS (NU SG)))) (AUX (TNS PRESENT) (VOICE
ACTIVE)) (VP (V COST) (OBJ (NP ($ 90))))) INTERP (FOR: THE A0013 /
(FINDQ: PERSON (FIRSTNAME BILL)) : T ; (FOR: THE A0014 / (FINDQ:
LOCATION (AMBIG WASHINGTON)) : T ; (FOR: THE A0015 / (FINDQ: DB/TRIP
(DESTINATION A0014) (TIME (AFTER NOW)) (TRAVELER A0013)) : T : (PROGN
(ADD: A0015 VALUE 90) (ADD: A0015 UNITS DOLLARS))))) INTERP (! THE
A0015 DB/TRIP ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
(TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
                                        **)
NIL NIL) SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))
```

```
(TIME (AFTER NOW)) (DESTINATION (! THE A0014 LOCATION ((AMBIG WASHINGTON))))
                                        *')
OBJ (NP ($ 90)) ACTION (PROGN (ADD: *** VALUE 90) (ADD: *** UNITS
DOLLARS)) SEMLINKS ((TIME (AFTER NOW)) (DESTINATION (! THE A0014
LOCATION ((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((
FIRSTNAME BILL))))) VOICE ACTIVE TNS PRESENT HEAD COST V COST SUBJ
(NP (DET (ART THE)) (N TRIP) (PP (PREP FOR) (NP (NPR (NAME (FIRST
BILL)))))) (PP (PREP TO) (NP (NPR (LOCATION (AMBIG WASHINGTON))))))
(FEATS (NU SG))) TYPE DCL SEMLINKS ((DESTINATION (! THE A0014 LOCATION
((AMBIG WASHINGTON)))) (TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL)))))
SEMLINKS ((TRAVELER (! THE A0013 PERSON ((FIRSTNAME BILL))))) SEMHEAD
DB/TRIP SEMVAR A0015 SEMQUANT THE)
States = END/ S/POP S/DCL-IS-ADV S/DCL-IS S/DCL-SUBJ-SG START/ S/
Scopedacts = (14)
Parent SC = 48

Creating constituent from Sconfig 49:

(FOR: THE A0013 / (FINDQ: PERSON (FIRSTNAME BILL))
      : T ; (FOR: THE A0014 / (FINDQ: LOCATION (AMBIG WASHINGTON))
                  : T ; (FOR: THE A0015 / (FINDQ: DB/TRIP
                                                  (DESTINATION A0014)
                                                  (TIME (AFTER NOW))
                                                  (TRAVELER A0013))
                              : T ; (PROGN (ADD: A0015 VALUE 90)
                                           (ADD: A0015 UNITS DOLLARS))))
      )

S DCL
  SUBJ NP DET ART THE
          N TRIP
          PP PREP FOR
             NP NPR NAME FIRST BILL
          PP PREP TO
             NP NPR LOCATION AMBIG WASHINGTON
          FEATS NU SG
  AUX TNS PRESENT
      VOICE ACTIVE
  VP V COST
     OBJ NP $ 90
```

SCONFIG 49 covers the entire sentence and will be recognized as a complete parse on the next page.

```
;  <GBROWN>APPENDIX-TRACE.SYNTAX;4    Mon 20-Dec-76 6:12PM  PAGE 1:24

Found complete parse:

     (FOR: THE A0013 / (FINDQ: PERSON (FIRSTNAME BILL))
          : T ; (FOR: THE A0014 / (FINDQ: LOCATION (AMBIG WASHINGTON))
                    : T ; (FOR: THE A0015 /
                                (FINDQ: DB/TRIP (DESTINATION A0014,
                                        (TIME (AFTER NOW))
                                        (TRAVELER A0013))
                                : T ;
                                (PROGN (ADD: A0015 VALUE 90)
                                       (ADD: A0015 UNITS DOLLARS)))))

     S DCL
       SUBJ NP DET ART THE
               N TRIP
               PP PREP FOR
                  NP NPR NAME FIRST BILL
               PP PREP TO
                  NP NPR LOCATION AMBIG WASHINGTON
               FEATS NU SG
       AUX TNS PRESENT
           VOICE ACTIVE
       VP V COST
          OBJ NP $ 90



Doing left proposals for Iconfig 11
     found 0 arcs reaching Iconfig.
Doing right proposals for Iconfig 11
     found 0 arcs reaching Iconfig.

Created new island:

Island config 11 = 0 BILL 1 -S 2 TRIP 3 TO 4 WASHINGTON 5 COSTS 6 NINETY
 7 DOLLAR-S 8
Left spansflag = DONE <-------------indicates a complete parse has been found.
Right spanstlag - T
Parent IC = 10
```

Official Distribution List

Contract N00014-75-C-0533

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314

Office of Naval Research
Information Systems Program
Code 437
Arlington, Virginia 22217

Office of Naval Research
Code 1021P
Arlington, Virginia 22217

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, Massachusetts 02210

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Illinois 60605

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, California 91106

New York Area Office
715 Broadway - 5th Floor
New York, New York 10003

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380

Office of Naval Research
Code 455
Arlington, Virginia 22217

Office of Naval Research
Code 458
Arlington, Virginia 22217

Naval Electronics Lab. Center
Advanced Software Technology Division
Code 5200
San Diego, California 92152

Mr. E. H. Gleissner
Naval Ship Research and
Development Center
Computation and Mathematics Dept
Bethesda, Maryland 20084

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D.C. 20350

Mr. Kin B. Thompson
Technical Director
Information Systems Division (OP-91T)
Office of Chief of Naval Operations
Washington, D.C. 20350

Advanced Research Projects Agency
Information Processing Techniques
1400 Wilson Boulevard
Arlington, Virginia 22209

Commanding Officer
Naval Air Development Center
Warminster, Pennsylvania 18974

Professor Omar Wing
Dept. of Electrical Engineering
Columbia University
New York, New York 10027

Assistant Chief for Technology
Office of Naval Research
Code 200
Arlington, Virginia 22217